VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

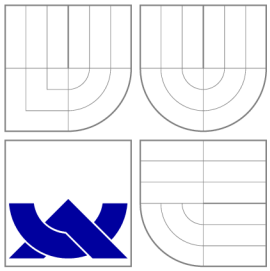
ROZŠÍŘENÍ NÁSTROJE PRO PODPORU AGILNÍHO VÝVOJE SOFTWARU

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

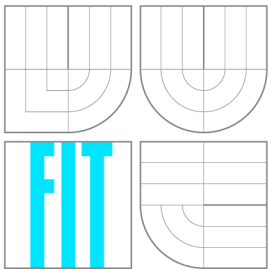
AUTOR PRÁCE
AUTHOR

Bc. PETR TRÁVNÍK

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ROZŠÍŘENÍ NÁSTROJE PRO PODPORU AGILNÍHO VÝVOJE SOFTWARU

UPGRADE OF AGILE DEVELOPMENT SUPPORT TOOL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PETR TRÁVNÍK

VEDOUCÍ PRÁCE
SUPERVISOR

doc. RNDr. JITKA KRESLÍKOVÁ, CSc.

BRNO 2014

Abstrakt

Cílem diplomové práce „Rozšíření nástroje pro podporu agilního vývoje softwaru“ je studium agilních metodik a jejich aplikace v praxi. Z agilních metodik se práce detailněji zabývá metodikou Scrum, kterou používá oddělení Corporate Technology společnosti Siemens v Brně. Práce se dále věnuje analýze a srovnání nejpoužívanějších profesionálních nástrojů pro agilní vývoj, které zároveň poskytují inspiraci pro rozšíření nástroje v oddělení společnosti Siemens. Na základně analýzy byla identifikována možná vylepšení nástroje s cílem ještě více zefektivnit agilní vývoj. Tyto závěry byly předloženy konzultantovi ze společnosti Siemens a na základě vzájemné dohody byly implementovány moduly pro *revizi kódu* a *retrospektivu*. Součástí implementace bylo také několik dílčích úprav současného nástroje. Všechna implementovaná rožšíření byla prováděna s důrazem na úsporu času, optimalizaci administrativní zátěže a další zefektivnění vývoje. Závěrem je diskutován přínos implementovaných řešení a možné další směry rozvoje nástroje.

Abstract

The goal of the diploma thesis „Upgrade of agile development support tool“ is to study agile methodologies and its use in practice. Thesis is focused on the Scrum framework used by The Corporate Technology department of Siemens, s.r.o. in Brno. Furthermore the thesis analyzes and compares the most used software support tools for agile methodologies which also gives an inspiration for the upgrade of support tool used by the department of Siemens. Thesis identifies possible upgrades based on an analysis with the goal to make agile development even more effective. Results were consulted with the representative of the Siemens company and then modules for Code review and Retrospective were chosen to implement. Implementation consists even of some minor upgrades of the tool. Goals of all implemented upgrades were to save time, optimize administrative work and make development even more effective. Benefits and further upgrades are consulted at the end of the thesis.

Klíčová slova

Agilní vývoj, Scrum, MVC, Team Foundation Server, MS Outlook, Vývoj software, Revize kódu, Retrospektiva, Kvalita kódu

Keywords

Agile development, Scrum, MVC, Team Foundation Server, MS Outlook, Software Development, Code Review, Retrospective, Code quality

Citace

Petr Trávník: Rozšíření nástroje pro podporu agilního vývoje softwaru, diplomová práce, Brno, FIT VUT v Brně, 2014

Rozšíření nástroje pro podporu agilního vývoje softwaru

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením doc. RNDr. Jitky Kreslíkové, CSc.
Další informace mi poskytl konzultant společnosti Siemens, s.r.o. Ing. Radek Gajdušek.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Trávník
27. května 2014

Poděkování

Rád bych poděkoval především konzultantovi práce Ing. Radku Gajduškovi za podporu, čas a úsilí mně věnované. Dále pak vedoucí práce doc. RNDr. Jitce Kreslíkové, CSc. za vedení, čas a úsilí.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Předmětem diplomové práce je rozšíření nástroje pro podporu agilního vývoje software v oddělení Corporate Technology společnosti Siemens, s.r.o. v Brně. Společnost praktikuje agilní vývoj pomocí metodiky Scrum a momentálně používá vlastní nástroj, který automatizuje velkou část procesů metodiky. Cílem práce je najít možnosti rozšíření nástroje a ještě více tak zefektivnit softwarový vývoj tohoto oddělení.

Úvodní kapitola se zabývá rozborem v současnosti používaných agilních metodik, jejich rozdíly a zásadami. Nejdříve se věnuje vzniku agilního vývoje jako metodologie a dále pak rozebírá extrémní programování, FDD, DSDM a důraz klade na Scrum, používaný ve společnosti Siemens.

Následující kapitola rozebírá současné profesionální nástroje pro podporu agilního vývoje. Především se zajímá o funkce a možnosti jednotlivých aplikací a nabízí jejich celkové srovnání. Pro rozbor jsou zvoleny nejpoužívanější nástroje dle aktuálního stavu na trhu.

V další kapitole se rozebírá nástroj aktuálně používaný v oddělení Corporate Technology společnosti Siemens. Věnuje se popisu jeho funkcí a identifikuje možná vylepšení, která by týmu napomohla odstranit další administrativu a přinést také časovou úsporu.

Kapitola pátá se již zabývá konkrétním návrhem a specifikací rozšíření nástroje. Specifikuje moduly pro skupinovou *revizi kódu* a dále pro správu *retrospektiv sprintů*. Kapitola popisuje také technologie a vývojové prostředí, které bude použito pro implementaci.

Další kapitola obsahuje popis implementace a testování. Lze v ní nalézt struktura implementovaných modulů a dalších dílčích vylepšení. Podkapitola testování představuje způsoby a konkrétní scénáře použité pro odhalování chyb.

V závěrečné kapitole je diskutován celkový přínos práce a další možné směry vývoje. Zhodnocení práce vychází z kvalifikovaných odhadů vedoucího oddělení a z reálného využití implementovaných modulů. Další směry vývoje se pak zaměřují na popis dalších možných rozšiřujících modulů.

Tato diplomová práce přímo navazuje na semestrální projekt vypracovaný v zimním semestru, který se zabýval agilními vývojovými metodikami, srovnáním profesionálních nástrojů pro podporu agilního vývoje a analýzou současného stavu na oddělení Corporate Technology v Brně. Uvedené kapitoly byly převzaty i do textu této práce.

Kapitola 2

Agilní vývoj

Agilní vývoj software je zastřešující pojem, pod něhož lze zařadit jednotlivé moderní vývojové metodiky software se společnými rysy [7]. Původně se jednalo o metodiky Scrum, XP, Crystal, FDD a DSDM, postupně se však přidávaly další. Společným rysem těchto metodik je iterativní vývoj, který je zvláště výhodný při spolupráci se zákazníkem.

2.1 Manifest

Pojem agilního softwarového vývoje byl definován v únoru roku 2001, kdy se sešlo několik předních odborníků na vývoj software. Výsledkem setkání byl dokument - manifest agilního vývoje. Manifest představil čtyři hlavní zásady a dvanáct principů z nich vycházejících. Signatáři manifestu dále založili neziskovou organizaci Agile Alliance, která propaguje a vytváří podporu pro standard agilního vývoje software.

Znění manifestu (čtyři hlavní zásady) zdůrazňuje priority při řízení vývoje [11]:

„Jednotlivci a interakce před procesy a nástroji.
Fungující software před vyčerpávající dokumentací.
Spolupráce se zákazníkem před vyjednáváním o smlouvě.
Reagování na změny před dodržováním plánu.“

2.2 12 principů

Ze čtyř hlavních zásad dále vyplývá dvanáct principů, které určují konkrétní požadavky na agilní metodiky.

1. Naší nejvyšší prioritou je uspokojení zákazníka a to formou průběžného dodávání produktu k otestování a případným změnám.
2. Akceptujeme požadavky na změnu a to i v pozdní fázi vývoje. Agilní metodiky využívají připravenosti na změny k získání konkurenčních výhod pro zákazníka.
3. Dodáváme funkční software v dostatečné časové periodě – od několika týdnů po několik měsíců – s preferencí kratší periody.
4. Vedení a vývojový tým musí spolupracovat denně napříč celým projektem.

5. Projekty staví na motivovaných zaměstnancích. Důraz musí být kladen na pracovní prostředí, podporu a důvěru tak, aby zaměstnanci bez problémů vykonávali svou práci.
6. Nejefektivnější a nejsmysluplnější metodou pro předávání informací týmu je konverzace z očí do očí.
7. Funkční software je primární metrikou pokroku.
8. Agilní procesy podporují neustálý vývoj. Investoři, vývojáři i uživatelé by měli být schopni neustále udržovat krok.
9. Důraz na technickou dokonalost a dobrý návrh zlepšuje obratnost projektu.
10. Jednoduchost minimalizuje nadbytečnou práci.
11. Nejlepší architektura, požadavky a návrhy přichází ze soběstačných týmů.
12. V pravidelných intervalech musí u týmu nastávat sebereflexe a snaha o zefektivnění. Závěry z těchto činností aplikovat pro své budoucí chování.

2.3 Extrémní programování

Extrémní programování bylo představeno v roce 1998 a zpopularizováno o rok později díky knize Kenta Becka Extreme Programming Explained: Embrace Change [5]. Metodika si v té době získala velkou oblíbenost především díky své netradičnosti a inovaci. Definice metodiky se pro stručnost shrnuje do dvanácti bodů:

Plánovací hra Na začátku každé iterace se sejdou zákazníci, manažeři a vývojáři, kteří navrhnou, odhadnou a určí priority požadavkům pro další iteraci. Požadavky se nazývají „uživatelské příběhy“ a zachycují se na „karty příběhů“ v jazyku dostupném všem stranám.

Krátké release První verze systému je vypuštěna do produkce již po několika iteracích. Od té doby se nasazuje do produkce nová pracovní verze každých několik dní až týdnů.

Metafora Zákazníci, manažeři a vývojáři vytvoří sadu „metafor“, podle kterých se modeluje výsledný systém.

Jednoduchý návrh Vývojáři jsou tlačeni k co nejjednodušším návrhům, aby ctili zásadu „vše jednou a pouze jednou“.

Testy Vývojáři před samotnou implementací nejprve napíší akceptační testy. Zákazníci zase pro každou iteraci navrhují funkcionální testy. Na konci iterace by měl produkt projít všemi testy.

Refaktorizace V průběhu vývoje by se měl návrh vyvíjet, aby byl stále udržován v co nejjednodušším stavu.

Párové programování Kód píší dva vývojáři sedící u jednoho počítače.

Kontinuální integrace Vývojáři integrují nový kód do systému, co nejčastěji je to možné. Všechny funkcionální testy musí po integraci projít, jinak jsou změny okamžitě zahozeny.

Kolektivní vlastnictví Kód je vlastněn všemi vývojáři najednou - mohou tedy dělat změny kdekoliv uznají za vhodné.

Přítomnost zákazníka Zákazník pracuje s vývojovým týmem kdykoliv je zapotřebí zodpovědět otázky nebo provést akceptační testy a zajistit tak, že vývoj probíhá podle plánu a očekávání.

40 hodinový týden Požadavky každé iterace musí respektovat 40 hodinovou týdenní pracovní dobu tak, aby vývojáři nemuseli pracovat přesčas. Metodika přesčasy výslovně zakazuje kvůli zvýšení výkonnosti týmu.

Otevřené pracoviště Vývojáři pracují na společném pracovišti. Individuální pracovní stroje jsou umístěny na okraji pracoviště a vývojové stroje uprostřed.

Jelikož musí být pracovní tým umístěn na jednom pracovišti, doporučuje se udržovat počet členů mezi dvěma až deseti vývojáři. Metodiku v důsledku požadavku na společný prostor nelze aplikovat na distribuované týmy. Maximální efektivita se dostaví pouze při dodržování všech principů.

Délka iterace je doporučena na dva týdny a jedná se tedy o jednu z nejkratších mezi agilními vývojovými metodikami.

2.4 Scrum

Vzhledem k dalšímu zaměření práce lze tuto podkapitolu považovat za stěžejní. Metodika je používána ve společnosti Siemens a pro výsledek této práce budou důležité především pojmy *revize kódu* a *retrospektiva*.

2.4.1 Historie

V roce 1986 se objevila zásadní studie pro budoucí vývoj produktových metodik. Studie s názvem „The new new product development game“ se snaží identifikovat důvody tržních úspěchů tehdejších korporací. Autoři Hirotaka Takeuchi a Ikujiro Nonaka v ní tvrdí, že zdrojem inovací a úspěchů nových produktů je především metodika jejich vývoje. Úspěšné korporace nepoužívají do té doby tradiční sekvenční vývoj (jako je například vodopádový model vývoje software), ale vytváří interdisciplinární týmy složené ze specialistů pokrývajících všechny potřebné obory. Všichni členové takových týmů spolupracují a iterativním způsobem vytváří nový produkt.

Na začátku devadesátých let se touto studií inspirovali průkopníci metodiky Scrum Jeff Sutherland a Ken Schwaber [14], kteří ji úspěšně nasadili ve svých softwarových firmách. V roce 2002 pak sepsali dokument „The Scrum Guide“ [12], který metodiku zpopularizoval a je průběžně revidován a překládán do mnoha světových jazyků. Momentálně se jedná o jednu z nejpoužívanějších agilních metodik vůbec (dle průzkumu [10]).

Jméno metodiky vychází z pojmu používaném v americkém fotbalu. V českém jazyce se Scrum překládá jako „mlýn“ -– stav, kdy se jednotliví hráči semknou a společně v jednom uskupení se snaží dosáhnout cíle, a sice dostat míč do soupeřova brankového pásma.

2.4.2 Základní definice

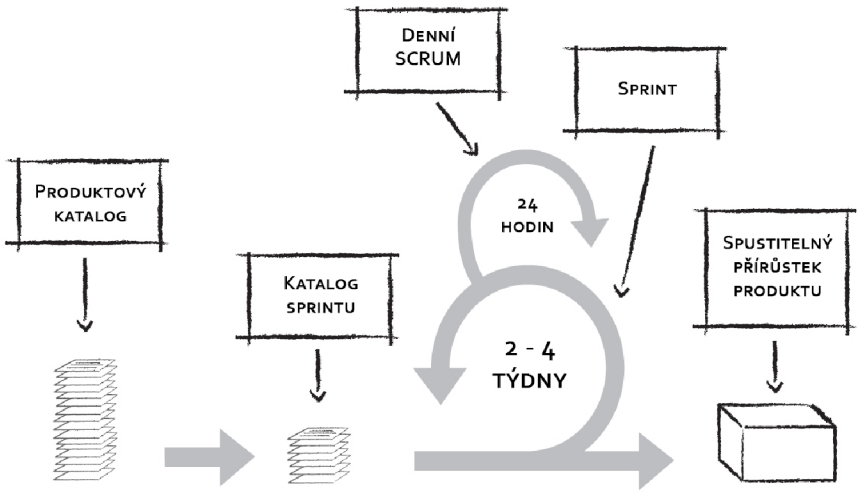
Software vyvíjený pomocí Scrum přináší výsledky ve dvou až čtyřtýdenních iteracích – *sprintech* (viz obrázek 2.1). Týmy se skládají ze tří až devíti členů (čerpáno z [13], počty se

v pramenech různí) zastupujících všechny pozice potřebné k vývoji software. Na začátku *sprintu* vždy proběhne plánovací schůzka *sprintu*, kde tým zvolí položky z *produktového katalogu*. Položky *produktového katalogu* jsou řazeny dle priorit. Tým se na základě zkušeností snaží co nejlépe odhadnout časovou náročnost implementace jednotlivých položek. Pro *sprint* pak volí takovou část *produktového katalogu*, jakou považuje za splnitelnou. Část katalogu zvolená pro jeden *sprint* se nazývá *katalog sprintu* a volí ho jen a pouze tým nezávisle na zainteresovaných stranách.

Při *sprintu* se pořádá *denní Scrum* — patnáctiminutová schůzka na začátku každého dne, kde se diskutuje pokrok v řešení *katalogu sprintu*, problémy na které členové narazili a na čem budou dále pracovat.

Výstupem *sprintu* je vždy spustitelný produkt s přírůstkem, který splňuje alespoň část *produktového katalogu*. Zainteresované strany (manažeři, ředitelé, akcionáři,...) pak mohou během *revize sprintu* zhodnotit současný stav produktu a případně ovlivnit *produktový katalog* přidáním, odebráním nebo změnou priorit jednotlivých položek.

Na závěr se tým ještě jednou sejde u *retrospektivy sprintu*, kdy zhodnotí práci týmu a především efektivitu a vytváří závěry ovlivňující vedení a průběh dalších *sprintů*.



Obrázek 2.1: Průběh iterace Scrum, inspirováno [14]

2.4.3 Artefakty

Tato podkapitola se zabývá prvky Scrum z předchozí kapitoly a *revizí kódu*. *Revize kódu* je důležitá pro následující kapitoly, a proto je jí věnováno více pozornosti.

Produktový katalog

Základním nástrojem pro správu požadavků je ve Scrum *produktový katalog*. Jedná se o seznam řazený pomocí priorit jednotlivých položek. Každá položka určuje nějakou funkci nebo vlastnost výsledného produktu. Příkladem položky *produktového katalogu* může být:

- Jako uživatel chci přidat online synchronizaci tak, abych mohl sdílet data se všemi lidmi dostupnými v mé síti.

- Jako administrátor chci přidat možnost filtrovat export tak, abych mohl exportovat položky pouze pro jednotlivé zákazníky.

Produktový katalog se mění v průběhu vývoje a právo editovat ho má pouze Vlastník produktu (role popsána v následující kapitole). Položky se mohou upravovat v případě nevhodné funkčnosti. Již nežádoucí položky můžeme z katalogu vymazat. Požadavky, které jsme dříve neočekávali, do *produktového katalogu* zapracujeme.

Pro určení priorit jednotlivých položek můžeme využít různé techniky dle povahy projektu (např. návratnost investice při zaměření na finanční dopad projektu). Položky nemusí být detailně popsány a ani by v první fázi neměly brát ohled na délky *sprintů*. Pokud je položka příliš náročná na jeden *sprint*, pak si tým po dohodě s Vlastníkem produktu zajistí rozdělení na dílčí položky.

Dle priorit jednotlivých položek katalogu se vytvoří pořadí a každé položce je přiřazeno unikátní *ohodnocení položky* udávající toto pořadí (z angl. *Stack Rank*). Dle těchto ohodnocení se pak vybírají položky do *katalogu sprintu*, které budou implementovány jako první.

Položky katalogu dále dostávají přiřazenu náročnost. Náročnost se udává jako tzv. *ohodnocení scénáře* (z angl. *Story points*). U většiny týmů se způsoby ohodnocení liší. Doporučuje se použití systému vlastních abstraktních stupnic — např. stupnice pro velikost oblečení — XS, S, M, L, XL.

Každý vývojový tým má jinou strategii, kdy považuje položku katalogu za hotovou. Tato strategie se označuje jako tzv. definice „hotovo“. Definice je vytvářena ve spolupráci zainteresovaných stran, vývojového týmu i vyššího managementu. Zainteresované strany například mohou požadovat dokumentaci každé hotové položky uloženou na svých serverech, vývojový tým může vyžadovat *revizi kódu* každé hotové položky katalogu, apod.

Sprint

Veškeré plánování u projektů využívajících metodiku Scrum se řídí pomocí *sprintů*. *Sprinty* jsou iterace o pevně dané délce dva až čtyři týdny.

Na začátku každého *sprintu* probíhá plánování, kdy tým zvolí položky *produktového katalogu* do *katalogu sprintu*. Položky je nutné volit s ohledem na výkonnost a dostupnost týmu. Každý *sprint* totiž obsahuje různý počet dostupných člověkohodin kvůli dovoleným, školením apod. Cílem *sprintu* je splnění všech položek z *katalogu sprintu*. V rámci *sprintu* se položky rozpracovávají shora dolů dle priority a tím je zajištěno splnění důležitých položek i při špatném naplánování *sprintu*. Pro sledování výkonnosti se často používají grafy se zobrazením již dokončené a zbývající práce. Tyto grafy jsou založené na hodinovém ohodnocení jednotlivých úkolů vzniklých rozdělením uživatelských scénářů, a tedy podávají přesné informace o aktuálním stavu.

Výstupem každého *sprintu* je spustitelná verze produktu s přírůstkem. Díky tomu mohou zainteresované strany průběžně reagovat na stav vývoje a flexibilně měnit požadavky. Používání průběžného dodávání produktu zákazníkovi snižuje riziko změnového řízení v pozdějších fázích vývoje a snižuje tak náklady.

Pro přírůstky produktu se opět využívá definice „hotovo“. Všechny položky *sprintu*, které spadají do přírůstku musí splňovat svou definici „hotovo“.

Revize kódu

Jednou z částí Scrum, která bývá mnohdy opomíjena je *revize kódu*. *Revize kódu* by se měly vyskytovat v dostatečné míře při provádění jednotlivých *sprintů*, jejich počet ale nemusí být přesně stanoven. Skupinová *revize* spočívá ve schůzce, na které členové vývojového týmu postupně prezentují a diskutují kód. Měla by mít předem danou délku, která zamezí zbytečnému plýtvání časem. Další formou pak může být použití nástroje pro online *revizi kódu*. Vývojář označí část kódu, kterou si není jistý a ostatní členové posílají své poznámky a návrhy řešení. Výběr opravy, která se nakonec použije pak může probíhat například hlasováním. Výhodou tohoto přístupu je možnost okamžité odezvy. *Revize kódu* tak přináší široký užitek:

- slouží ke zkvalitnění výstupu
- předání zkušeností mezi členy týmu navzájem.
- skupinová *revize* nutí zaměstnance psát čitelný a strukturovaný kód, za který se nebudou stydět při prezentaci
- zajišťuje se také integrita standardů pro formátování a zápis kódu
- chyby mohou být díky *revizi* odhaleny dříve, než se stanou kritickými

Úseky určené k *revizi* se mohou vybírat podle různých kritérií:

- kritické části kódu s vysokou prioritou
- úsek psaný novým zaměstnancem
- složitý kód např. spojující různá rozhraní

Schopnost odstraňovat chyby v co nejranější fázi je při softwarovém vývoji kritická. Některé týmy volí cestu striktních procesů, párového programování nebo píší vícenásobně testů jako samotného kódu. Většina z těchto technik ale není příliš motivující, případně není vhodná pro distribuované týmy. *Revize kódu* přináší univerzální řešení, které navíc podporuje rozvoj zaměstnanců a jejich motivaci.

Revize kódu může znamenat velké množství administrace ať už při organizování nebo při sledování oprav a možná proto bývá často opomíjena. Výhodnou strategií je tak volba podpůrného software, který automatizuje všechny tyto úkoly a umožní vývojářům soustředit se čistě na svou práci.

Důležitým aspektem při provádění *revizí* je zachování koncepce skupinového vlastnictví kódu. Pokud se stanou *revize* spíše nástrojem pro hledání chyb ve vývojářích, než v samotném kódu, pak mohou mít negativní dopad na celý vývojový tým a narušit jeho spolupráci. Vývojáři tedy musí být vedeni, aby vnímali *revizi* jako nástroj pro získání zkušeností a nových znalostí. Zavedení metrik může být obtížné z hlediska velkého množství nepřímých vlivů *revize kódu* na vývojové procesy. Pro měření úspěšnosti tak zůstavají metriky jako počet nalezených chyb, jejich kritičnost apod.

2.4.4 Role

Metodika Scrum zavádí do vývoje několik různých rolí, které v rámci projektu spolupracují. Scrum tým se skládá z vývojářů a dále dvou speciálních rolí — Mistr Scrum a Vlastník produktu. Mezi zainteresované strany dále patří zákazníci a zainteresované strany, kteří výsledný produkt používají, resp. financují.

Scrum mistr

Scrum mistr je klíčová role pro úspěšné používání metodiky Scrum. Nejedná se o tradiční vedoucí nebo manažerskou pozici a roli může zastávat i vývojář. Scrum mistr především kontroluje a řídí používání technik metodiky Scrum. Konkrétně má za úkol:

- kontrolovat pokrok
- zajišťovat odstraňování překážek
- řídit plánování, *revize* a *retrospektivy sprintů*, *denní Scrum*, atd. - obecně vše, co souvisí s metodikou Scrum
- zajišťovat podporu pro průběžné zlepšování procesů
- pomáhat komunikaci mezi zainteresovanými stranami a týmem

Vývojáři

Vývojáři jsou členové týmu, kteří zajišťují samotný návrh, implementaci, testování a tvorbu dokumentace. Z pohledu metodiky Scrum nejsou určeny žádné role pro jednotlivé členy týmu a všichni tak dělají vše. V týmu se mohou objevit specialisti, kteří mohou pomoci ostatním členům, ale práce je rozdělována všem rovnoměrně bez rozdílu.

Vlastník produktu

Vlastník produktu je zodpovědný za určení a komunikaci vize produktu a určení priorit pro jeho funkčnost. Konkrétně to znamená následující úkoly:

- správa *produktového katalogu*
- stanovení ohodnocení položek *produktového katalogu* (určuje priority)
- sjednocení vize mezi vývojáři a zainteresovanými stranami
- určuje, co se má udělat a v jakém pořadí
- vytváří plány nasazení a jejich termíny
- zajišťuje podporu pro plánování *sprintů* a *revizí*
- reprezentuje zákazníky a zainteresované strany

Vlastník produktu by měl být k dispozici vždy, když narazí vývojáři na nejasnosti v *produktovém katalogu*. Vlastník se řadí do Scrum týmu.

Zainteresované strany

Role, která se již neřadí do Scrum týmu. Komunikují výhradně s Vlastníkem produktu, který jim tak zajišťuje roli prostředníka při komunikaci s vývojovým týmem. Mají vliv při určování vize a prostřednictvím Vlastníka produktu ovlivňují *produktový katalog*.

2.5 Metodika vývoje dynamických systémů

Z anglického Dynamic systems development method, dále jen DSDM. Je založeno na devíti klíčových principech. Ty se dotýkají business potřeb, aktivního zapojení uživatele, motivovaného týmu, častého vydávání nových verzí, integrovaného testování a spolupráce zainteresovaných stran. DSDM považuje za primární kritérium dodání a přijetí systému součinnost s business účelem produktu. V souvislosti se zaměřením na business staví DSDM do popředí pravidlo „80% užitečnosti produktu je vytvořeno za 20% času“.

Správa a výběr požadavků pro aktuální iteraci probíhá pomocí MoSCoW priorit:

M – Must have požadavky, které musí být splněny

S – Should have požadavky, které by měly být splněny, pokud je to možné

C – Could have požadavky, které by se hodily, ale nejsou kritické

W – Won't have požadavky, které se prozatím implementovat nebudou, možná později

Pro každou iteraci by mělo být určeno několik kritických i nekritických požadavků. Kritické požadavky musí být vždy splněny a zbylý čas je určen pro nekritické.

DSDM standard je neustále vyvíjen a jeho aktuální verze pochází z roku 2007. DSDM rámec je možné aplikovat ve spojení s extrémním programováním.

2.6 Vývoj řízený funkčností

Z anglického Feature–driven development, dále jen FDD. Projekt je rozdělen na jednotlivé funkce a vlastnosti a tyto jsou poté implementovány jednotlivě. Na začátku vývoje se vytvoří model systému s vysokou úrovní abstrakce. Dále se pokračuje dvoutýdenními iteracemi, kdy se vždy vytvoří návrh a implementuje jedna z funkcí systému. Jednotlivé funkce by měly být dostatečně malé, ale užitečné v očích zákazníka. Při vývoji se používá 8 základních principů:

- návrh doménových objektů
- vývoj po funkcích
- vlastnictví tříd/komponent
- týmy funkce
- kontrola
- konfigurační management
- pravidelný překlad
- viditelnost pokroku a výsledků

U FDD si vývojář nevybírá, kterou funkci chce implementovat, ale je mu přidělena. Pokud je potřeba spolupráce několika vývojářů, zavádějí se podtýmy -— týmy funkce. Vývojový tým by měl využívat pravidelné sestavování hotových částí projektu. Důležitým prvkem FDD je také revize kódu, která zajišťuje sdílení zkušeností jednotlivých členů týmu. FDD je dle výsledků z praxe vhodný také pro větší týmy na rozdíl od Scrum nebo extrémního programování.

2.7 Crystal

Metodika byla představena v knize „Crystal Clear: A Human-Powered Methodology for Small Teams“ [6]. Napsal ji Allistair Cockburn, odborník na agilní metodologie a zároveň autor metodiky Crystal. Po představení metodiky vzniklo ještě několik dalších variant a Crystal tak momentálně označuje celou rodinu metodik, do které patří například *Crystal Clear*, *Crystal Yellow* nebo *Crystal Orange* (více v článku [8]).

Crystal je specifický především svou variabilitou a zastává názor, že každý projekt je unikátní a vyžaduje různé procesy pro svůj vývoj. Při používání metodiky Crystal tak při každém novém projektu měníme procesy a praktiky tak, aby byla zajištěna maximální efektivita. Metodika pak omezuje ze svého pohledu hlavní faktory vývoje jako priority, velikost týmu a sledování průběhu.

Důraz klade na týmovou součinnost, komunikaci, jednoduchost a častou sebereflexi, která má zajistit průběžnou optimalizaci procesů. Stejně jako ostatní agilní metodiky zdůrazňuje časté dodávky prototypů zákazníkovi již od prvních fází vývoje. Vývojový tým pak má být co nejlépe odstíněn od administrativních úkonů a dalších rozptýlení.

2.8 Kanban

Kanban je agilní metodika zaměřující se na implementační fázi. Zajišťuje proces rozdělování práce s cílem nepřetěžovat tým. Postupnou optimalizací faktorů jako maximální množství aktuálně rozpracovaných úkolů zajišťuje maximální efektivitu týmu ve fázi implementace.

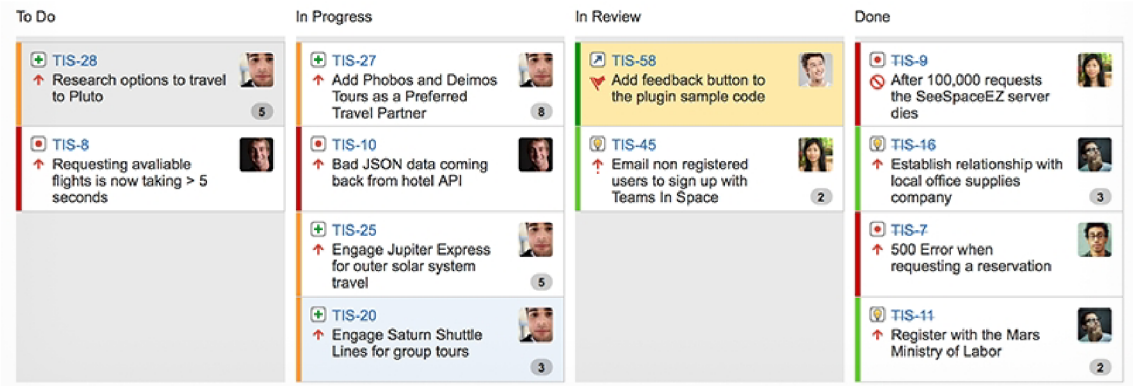
Je založen na třech principech:

Zobraz, co se dnes bude dělat - vývojový tým by měl vidět kontext aktuálního průběhu vývoje

Omez množství úkolů, na kterých se v jedné chvíli pracuje - zamezuje týmu, aby přeceňoval své síly a dostal se do stavu, kdy nestíhá

Dodržuj priority úkolů - když je jeden úkol dokončen, zvol další s nejvyšší prioritou

Jelikož Kanban nepokrývá veškeré fáze vývoje software, používá se ve spojení s dalšími agilními metodikami. Populární je například Scrum, který využívá Kanban pro průběh *sprintů*. Metodiku Kanban využívá také oddělení Corporate Technology.



Obrázek 2.2: Ukázka Kanban nástěnky v nástroji JIRA, zdroj: [3]

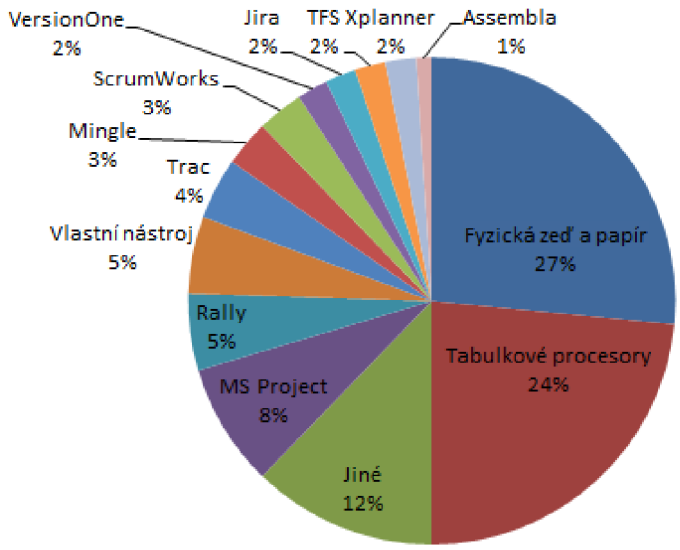
Kapitola 3

Nástroje pro podporu agilního vývoje

Následující kapitola se zabývá zavedenými nástroji pro podporu agilního vývoje. Zkoumá především jejich funkčnost a užitečnost tak, aby poskytla inspiraci pro návrh rozšíření nástroje, který je předmětem této práce.

3.1 Nejpoužívanější nástroje

V současné době existuje na trhu nepřeberné množství produktů od malých i velkých, začínajících i zavedených společností. Analýza nástrojů se zaměřuje především na podporu metodiky Scrum, která je pro práci stěžejní. Produkty k analýze byly zvoleny dle výsledků průzkumu trhu na obrázku 3.1 (zdrojem výzkum konaný v 35 zemích světa [10]).



Obrázek 3.1: Používanost agilních vývojových nástrojů, zdroj dat: [10]

Z analýzy byly vyloučeny nástroje, které nejsou určeny primárně pro agilní vývoj (MS Project, tabulkové procesory a Trac). Dále byl vyloučen projekt Xplanner, který byl odkoupen společností Atlassian a zaintegrován do nástroje JIRA.

3.2 JIRA

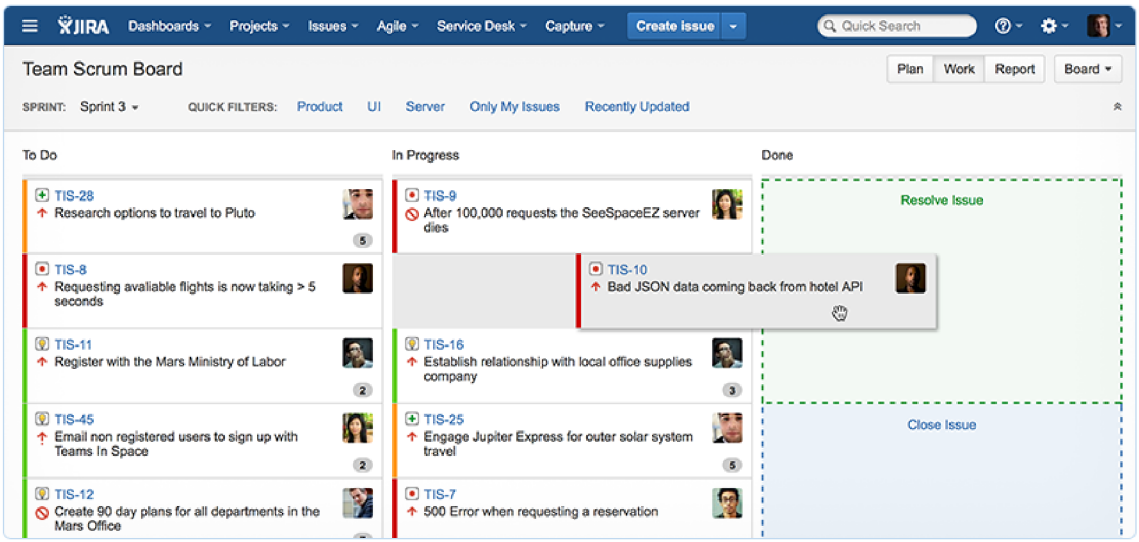
JIRA patří mezi nejoblíbenější nástroje pro podporu agilního vývoje. Nabízí širokou podporu pro různé metodiky vývoje a dodatečné integrovatelné nástroje například pro *revizi kódu*. Nástroj se na trhu drží už od roku 2002, což značí pevné zázemí produktu. Vyvíjen je australskou společností Atlassian, jejíž portfolio obsahuje produkty pro vývoj, nasazení a správu software [3].

Jedná se o webový nástroj přístupný přes prohlížeč, provozovaný v závislosti na licenci na vlastních serverech uživatele nebo hostovaný poskytovatelem. Vytváří podporu pro projektový management, správu defektů a požadavků. Spolu s rozšířením Agile pak nabízí správu projektů dle metodiky Scrum nebo Kanban.

Hlavní obrazovkou při vývoji pomocí Scrum se stává tzv. *Scrum Board*. Slouží pro zobrazení všech úkolů aktuálního *sprintu* a jejich stavů. Členové týmu tím získávají přehled o celkovém stavu *sprintu* a získají tak možnost efektivnějšího rozhodování při dělení práce. Na obrázku 3.2 vidíme stav, kdy prozatím žádný z úkolů nebyl dokončen. Existuje úkol s nekritickou prioritou, který je rozpracován, přitom ale stále zůstává úkol s kritickou prioritou, na kterém se nezačalo pracovat. Pro tým tento stav představuje motivaci ke zvýšení úsilí a nutnost přerozdělení úkolů.

JIRA dále poskytuje rozsáhlou podporu pro správu *produktového katalogu* a hodnocení priorit. Tým má díky nástroji k dispozici graf výkonnosti týmu v aktuálním *sprintu*. V neposlední řadě poskytuje také podporu pro správu *retrospektiv sprintů*. Strukturu *retrospektiv* má na starosti samotný tým a může tak aplikovat vlastní metody pro zápis.

JIRA nabízí také rozšíření pro *revizi kódu* s názvem Crucible. *Revize kódu* probíhá na základě požadavků autora. Zajišťována je následně jednotlivými členy týmu dle jejich dostupnosti. Nástroj spravuje statistiky jako počet nalezených chyb na *revizi* a procento revidovaného kódu. Výhodou je také podpora pro distribuované týmy.



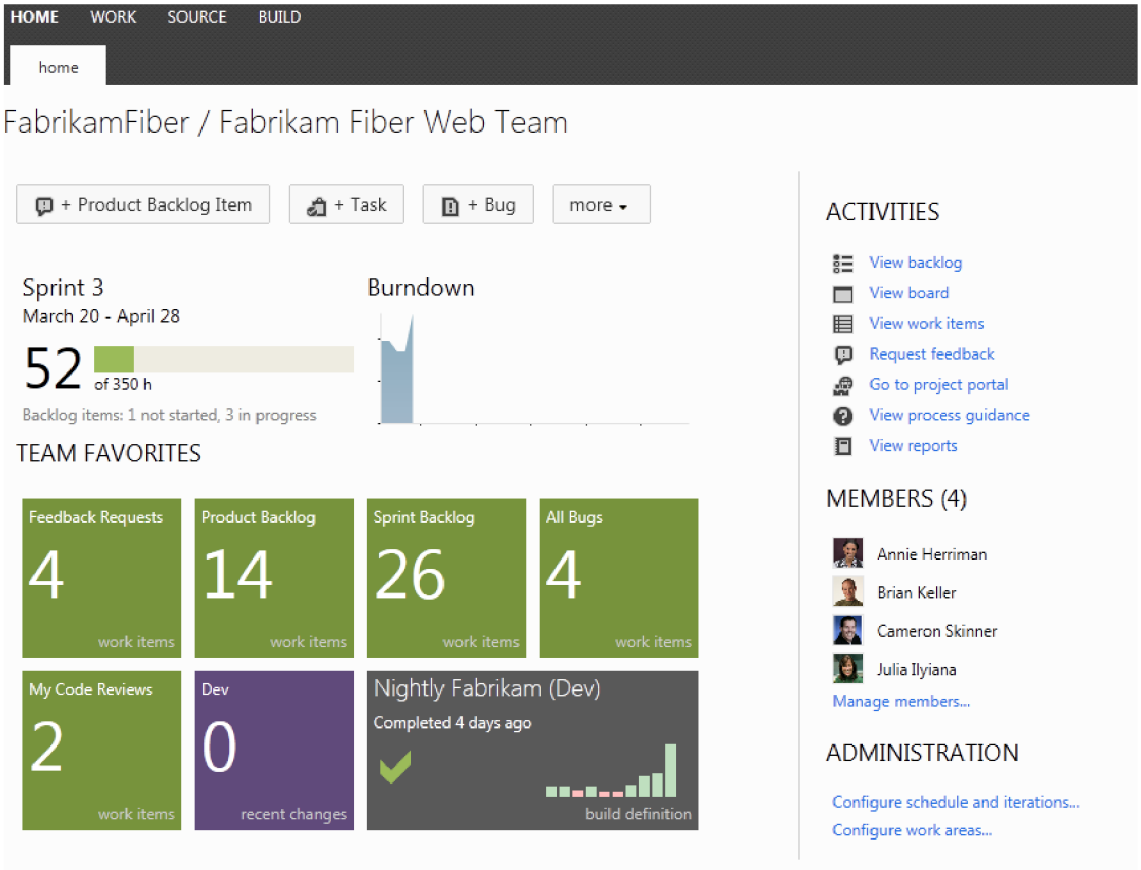
Obrázek 3.2: Úvodní obrazovka v nástroji JIRA, zdroj: [3]

3.3 Team Foundation Server

Dále jen TFS je produkt společnosti Microsoft, který nabízí podporu pro správu zdrojového kódu, správu požadavků, projektový management (pro agilní i vodopádové modely vývoje), testování, nasazení a mnohé další. Nástroj je určen především pro použití s vývojovými prostředími Visual Studio nebo Eclipse. TFS podporuje propojení s nástroji Microsoft Project, Excel a Powerpoint pro pohodlnou správu [4].

Pro přístup používá TFS webové rozhraní nebo přístup přímo z vývojového prostředí (např. Visual Studio). Základní obrazovku při používání metodiky Scrum můžeme vidět na obrázku 3.3. Výhodou nástroje je rozsáhlá správa *produktového katalogu*, která mimo jiné umožňuje distribuci mezi více týmů, čímž odstraňuje nevýhodu metodiky Scrum v omezeném počtu členů týmu. *Produktový katalog* je používán hierarchicky a velké položky, které přesahují rozsah *sprintu*, se mohou dělit až na úrovni týmů, přitom pro Vlastníka produktu zůstávají konzistentní.

Na úrovní týmu nástroj zobrazuje počty odpracovaných a dostupných člověkohodin, které navíc zobrazuje v grafu výkonnosti týmu. Díky těmto kapacitním informacím tak mohou týmy lépe plánovat *katalogy sprintů*. TFS poskytuje také grafické zobrazení stavu jednotlivých úkolů – podobně jako v předchozím nástroji JIRA. Testování kódu je integrováno mj. do webového rozhraní a členové týmu tak mohou spouštět testy přímo přes prohlížeč pomocí nástroje Manažer testů.



Obrázek 3.3: Úvodní obrazovka Team Foundation Server 2012 při využití Scrum, zdroj: [4]

Po skončení *sprintu* nabízí nástroj pro podporu *revize sprintu* a *retrospektivy*. K *retrospektivám* lze využít hodnocení *sprintů*, které mohou jednotliví členové vytvářet sami přímo v nástroji. Škálovatelnost TFS zajišťuje, že záleží pouze na týmu, kde bude zveřejňovat závěry z *retrospektiv*. Díky automatickému překladu jsou ihned dostupné přírůstky *sprintu* a mohou být okamžitě doručeny zákazníkovi.

Revize kódu probíhá v nástroji individuálně. Autor kódu označí část, která potřebuje *revizi* a následně ji provede některý z dostupných členů týmu, případně i skupina. Nalezené chyby jsou automaticky sledovány ve správě defektů.

TFS mimo jiné používá také oddělení Corporate Technology společnosti Siemens v Brně. Bohužel je ale dostupné pouze ve verzi 2010, která nepodporuje Scrum a z tohoto důvodu si společnost vytváří svůj vlastní nástroj, který využívá API, které TFS nabízí. Aktualizace verze TFS na oddělení by mohla přinést nativní podporu, jejíž výhodou by bylo přímé propojení s vývojovým prostředím.

3.4 ScrumWorks Pro

ScrumWorks Pro patří také do desítky nejrozšířenějších nástrojů pro podporu agilního vývoje. Již z názvu vyplývá specializace na metodiku Scrum. Nástroj je vyvíjen již od roku 2000 a nabízí jak desktopového klienta, tak přístup přes webové rozhraní [9].

Desktopový klient na obrázku 3.4 a webové rozhraní se ve funkčnosti liší, což způsobuje omezení možností pro externí přístup k nástroji – např. *produktový katalog* nebo tým je možné založit pouze z desktopové verze. ScrumWorks usnadňuje komunikaci týmu díky možnosti psaní komentářů členy týmu ke většině položek, které nástroj spravuje.

Stejně jako předchozí nástroje nabízí graf výkonnosti týmu tentokráte dostupný na kterékoliv obrazovce nástroje. Základní obrazovka se stavy úkolů aktuálního *sprintu* je v nástroji graficky rozdělena a položky katalogu jsou dále rozděleny na stavy jednotlivých scénářů úkolu. Nástroj tak nabízí detailní zobrazení stavu práce.

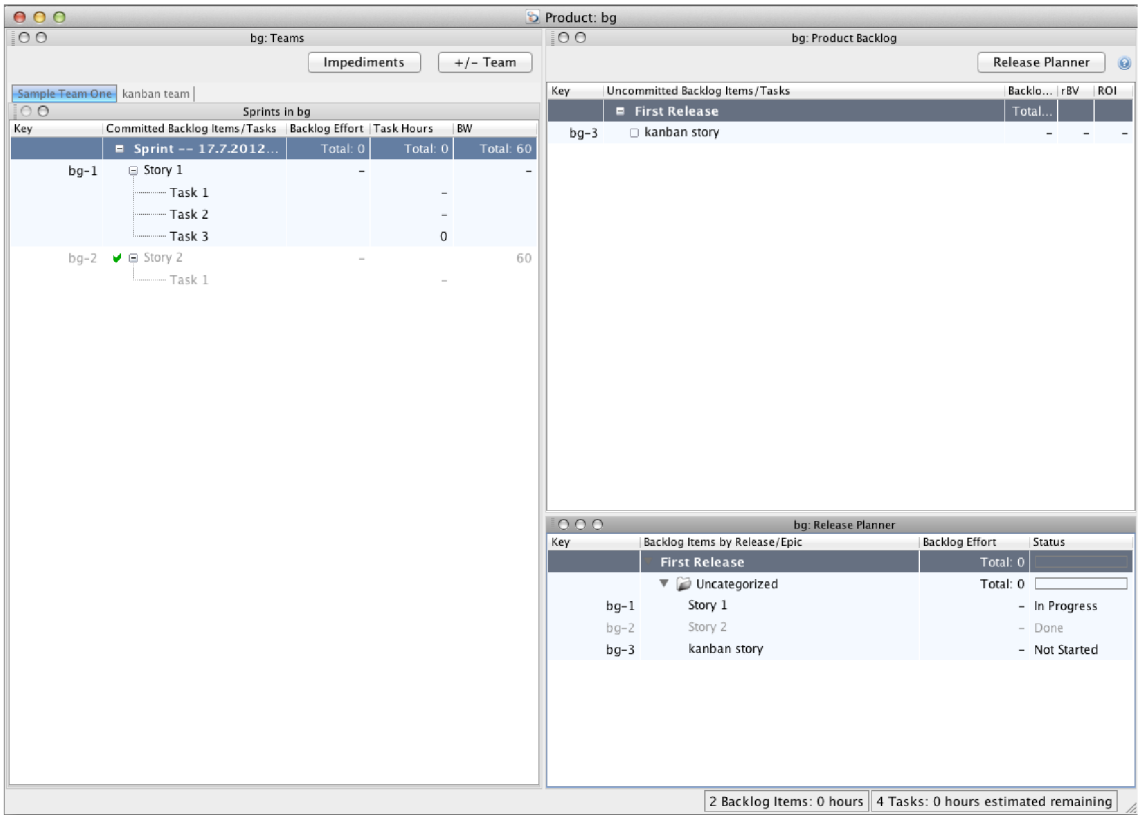
Nástroj zdůrazňuje účast vlastníka produktu na projektu. Jednotlivé scénáře v nástroji vždy musí uzavírat až Vlastník produktu, což zajišťuje průběžnou kontrolu práce. Na rozdíl od ostatních nástrojů navíc Vlastník produktu může určovat priority *produktového katalogu* pomocí business ukazatelů jako ROI, což se přibližuje agilní metodice DSDM.

3.5 Rally

Společnost, která vyvíjí tento nástroj - Rally Software patří mezi jedny z největších společností zabývajících se agilním vývojem. Založena byla roku 2001 a kromě podpůrných nástrojů poskytuje také služby jako agilní koučování včetně vlastní certifikace zaměřené na postupy zavádění agilních metodik [1].

Rally je webový nástroj, jehož výhodou je zvýšená podpora pro multioborové týmy. Počítá s různými obory členů vývojového týmu a nabízí možnosti pro řízení životních cyklů jednotlivých úkolů. Nástroj nabízí také integrovanou podporu managementu chyb. Částečnou nevýhodou je přílišná komplexita, která není doplněna dostatečně kvalitním uživatelským rozhraním. Problém také nastává při spravování portfolií, kde nástroj nevytváří dostatečnou podporu pro spojení vrstev projektů a portfolia.

Nástroj poskytuje přednastavené procesy pro agilní vývoj s velmi malou možností konfigurace (narozdíl například od VersionOne, který poskytuje konfigurovatelné šablony). Společnost, která nasadí tento nástroj, se musí přizpůsobovat nástroji a ne naopak. Toto ale



Obrázek 3.4: Desktopový klient nástroje ScrumWorks Pro, zdroj: [9]

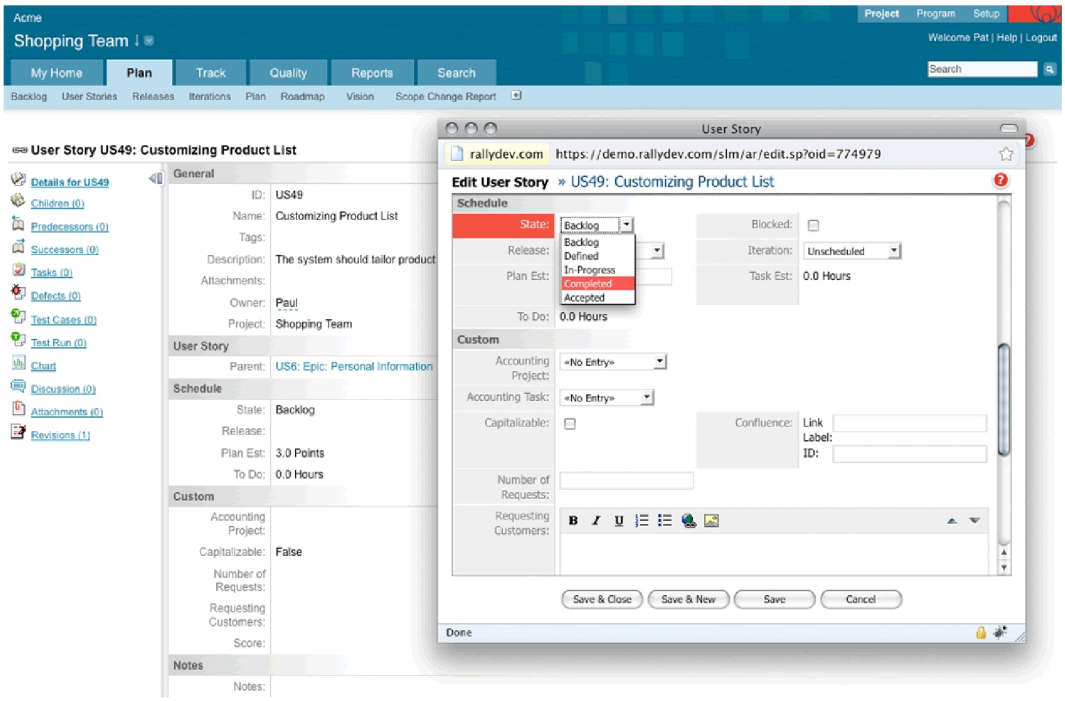
může představovat výhodu pro společnosti, které agilní vývoj teprve zavádí, jelikož nemusí definovat vlastní systém, ale jsou nástrojem vedeni. Rally neobsahuje žádný integrovaný nástroj pro správu *revizí kódu*.

3.6 Mingle

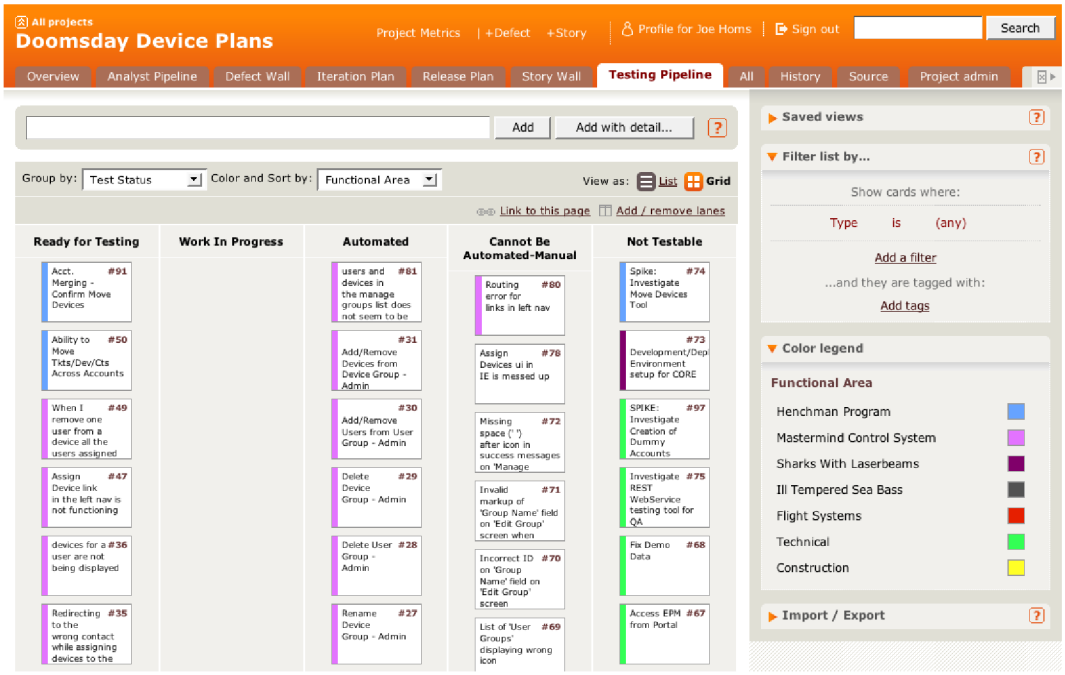
Společnost ThoughtWorks, která vyvíjí nástroj Mingle byla založena již v roce 1993. Společnost ovládá široké portfolio produktů, do kterého patří například software pro zprávu lékařských záznamů, rozšíření pro vývojové nástroje apod. [15].

Mingle je opět webový nástroj, který nabízí výběr mezi hostovanou a lokální verzí. Součástí software je také pokročilá správa defektů. V rozšířené verzi pro velké organizace nabízí Mingle podporu pro automatické upozorňování správců portfolia při nedostatečném pokroku v řešení dílčích úkolů apod. Nástroj klasicky zobrazuje grafy výkonnosti týmu navíc ale dokáže předpovídat průběh výkonnosti z historických dat.

Tento produkt podporuje vývoj pomocí Scrum, extrémního programování i dalších agilních metodik. Nástroj nabízí možnosti individuálních úprav šablon a je tak vhodný i pro organizace, které již agilní vývoj praktikují. Mingle klade důraz na funkci *karetní zdi* vycházející z Kanbanu, která slouží jako místo pro sdílení aktuálního průběhu *sprintu*. Jedná se o funkci, která umožňuje efektivní komunikaci také v distribuovaných týmech. Nástroj implicitně nepodporuje správu *revize kódu*.



Obrázek 3.5: Ukázka uživatelského scénáře v nástroji Rally, zdroj: [1]



Obrázek 3.6: Ukázka nástěnky nástroje Mingle, zdroj: [15]

3.7 VersionOne

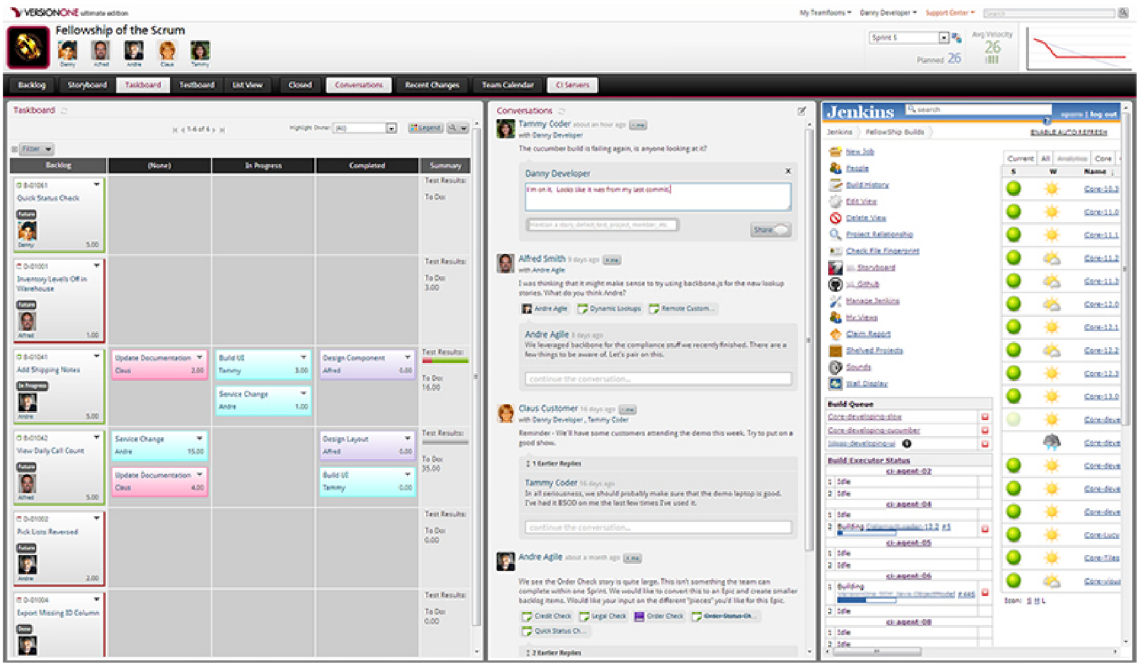
Historie stejnojmenného vývojáře nástroje VersionOne se píše od roku 2002 a patří tak mezi jednu z nejstarších společností zabývajících se pouze agilním vývojem. Společnost má

velmi pestrou zákaznickou základnu a působivé jsou především velké korporace (Lockheed Martin, SAP, Oracle nebo Nasdaq) [16].

Nástroj má nejširší spektrum podporovaných agilních metodik ze všech analyzovaných nástrojů. Kromě Scrum podporuje také Kanban, Crystal, FDD, XP a DSDM. Nástroj poskytuje velkou flexibilitu a pro uživatele není problém konfigurovat procesy přesně dle vlastních potřeb.

Agilní vývoj podporuje nástroj řadou modulů, z nichž zajímavý je například *TeamRoom* na obrázku 3.7. Jedná se o ucelený pohled na aspekty vývoje, které se týkají každodenních aktivit (aktuální stavy katalogů, nástěnka úkolů, grafy výkonnosti týmu, apod.). Pohled je plně konfigurovatelný a tým si tak může vybrat, které nástroje jsou pro něj nejdůležitější. Výhodné se tak jeví zobrazení tohoto modulu na pracovišti na viditelné obrazovce, kterou má celý tým v průběhu práce na očích.

Dalším zajímavým modulem je *PlanningRoom* určený pro plánování a správu *produktových katalogů*. Modul nabízí pohledy pro management, zainteresované strany i samotný tým. Dalšími zajímavými nástroji jsou například týmová sdílená nástěnka s podporou kreslení diagramů nebo zabudovaný chat. Pro *revizi kódu* nabízí nástroj možnost integrace externích nástrojů. Nástroj je vhodný pro společnosti, které mají dostatečné odhodlání pro agilní vývoj a potřebný kapitál, jelikož konfigurace vyžaduje množství zkušeností a ve většině případů také zaměstnání odborného konzultanta.



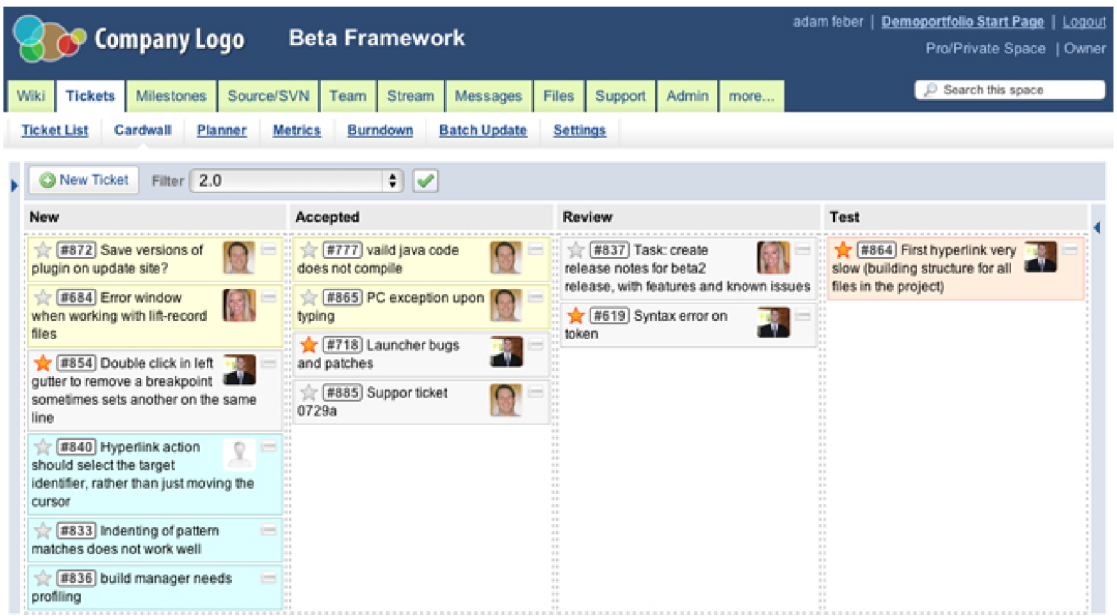
Obrázek 3.7: Modul TeamRoom nástroje VersionOne, zdroj: [16]

3.8 Assembla

Společnost Assembla na trhu působí od roku 2005. Společnost původně nabízela nástroje pro správu kódu a defektů. Později ale rozšířila portfolio také o produkty pro management agilního vývoje.

Assembla nabízí vlastní agilní metodiku vzniklou spojením Scrum a Kanban. Metodika je vhodná pro týmy méně zkušené v agilním vývoji, nástroj totiž nenabízí příliš prostoru pro vlastní úpravy procesů.

Produkt podporuje správu kódu a nabízí integrovaný systém pro *revizi kódu*. Systém funguje na bázi návrhů a hlasování. Uživatelé nejprve označí kód k *revizi* (případně se reviduje veškerý napsaný kód) a následně jsou ostatními uživateli navrhnuty opravy. Opravy jsou posléze ostatními uživateli odhlasovány jako výhodné, případně zamítnuty jako nevyhovující. Počet hlasů nutných pro přijetí nebo zamítnutí je konfigurovatelný a záleží již na týmu, jakou zvolí taktiku.



Obrázek 3.8: Kanban nástěnka nástroje Assembla, zdroj: [2]

3.9 Srovnání nástrojů

Veškeré analyzované nástroje uvádí podporu pro organizace s 1000 i více zaměstnanci. Většina nástrojů nabízí dva způsoby nasazení - hostované přímo u výrobce nástroje nebo lokální na serverech uživatele. Ceny produktů se velice liší a záleží především na poskytované podpoře ze strany výrobce produktu (opravy chyb, záchrana dat, školení, apod.).

Všechny analyzované nástroje podporují webové rozhraní, které se jeví jako nejvýhodnější pro tento typ nástroje. TFS a ScrumWorks navíc nabízí desktopové klienty, kteří se ale od webového rozhraní liší v množství funkcí. Pro malé společnosti jsou rozhodně výhodnější verze hostované, které umožňují oproštění od administrativy serveru a navíc stálou podporu. Nejlevnějšími řešeními jsou nástroje JIRA a Assembla, z nichž JIRA nabízí širší konfigurovatelnost. Konkrétní data k možnostem nasazení, typech klientů a ceny za licence se nacházejí v tabulce 3.2.

Jak je vidět na tabulce 3.3, nejuniverzálnějším nástrojem z výše analyzovaných se ukázal VersionOne, který podporuje nejvíce agilních metodik a navíc nabízí další možnosti konfigurace procesů. Všechny nástroje podporují Scrum alespoň v nějaké jeho formě. Vítaným rozšířením Scrumu je Kanban, který umožňuje zpřehlednění aktuálně prováděné práce.

Nástroj	Označení verze	Měsíc vydání
JIRA	6.1.5	prosinec 2013
TFS	2013	listopad 2013
ScrumWorks	6.2.0	listopad 2013
Rally	bez označení	prosinec 2013
Mingle	13.4	srpen 2013
VersionOne	Summer 2013	září 2013
Assembla	1.6 beta	prosinec 2013

Tabulka 3.1: Verze nástrojů použité k testování

Nástroj	Klient	Nasazení	Cena (uživatel/rok)
JIRA	webový	hostovaný/lokální	10$ – 25$
TFS	webový/desktopový	hostovaný/lokální	499$ neomezeně
ScrumWorks	webový/desktopový	hostovaný/lokální	276$ – 300$
Rally	webový	hostovaný	420$ – 588$
Mingle	webový	hostovaný/lokální	240$ – 400$
VersionOne	webový	dle verze	109$ – 468$
Assembla	webový	hostovaný/lokální	13$ – 20$

Tabulka 3.2: Srovnání nástrojů dle možností nasazení, klientů a ceny

Vysvětlivky:
✔ – nativní podpora
➞ – nutno instalovat rozšíření

Nástroj	Scrum	Kanban	XP	FDD	DSDM	Crystal	Lean	RUP
JIRA	✔	✔						
TFS	✔							
ScrumWorks	✔	✔					✔	
Rally	✔		✔					✔
Mingle	✔		✔				✔	
VersionOne	✔	✔	✔	✔	✔	✔		
Assembla	✔	✔						

Tabulka 3.3: Podpora metodik

Všechny analyzované nástroje nabízejí ať už méně či více širokou podporu pro správu *produktového katalogu* a kvalitní graf výkonnosti týmu. Pokud nabízel nástroj *revize kódu*, pak se většinou jednalo o rozšíření nutné zakoupit zvlášť. Překvapením byla nedostatečná nativní podpora pro *retrospektivu* a *revizi sprintu*, kterou můžeme vyčíst z tabulky 3.4.

Nástroj	Produktový katalog	Retrospektiva	Revize sprintu	Graf výkonnosti	Revize kódu
JIRA	✔	✔	✔	✔	→
TFS	✔	✔	✔	✔	✔
ScrumWorks	✔			✔	
Rally	✔		✔	✔	
Mingle	✔	✔		✔	
VersionOne	✔	✔	✔	✔	→
Assembla	✔			✔	✔

Tabulka 3.4: Podpora artefaktů Scrum

Kapitola 4

Analýza současného nástroje

Současný nástroj pro podporu agilního vývoje software vznikl ve společnosti Siemens, s.r.o. také jako diplomová práce, jejímž autorem je Ing. Radek Gajdušek. Nástroj vytváří jednotné prostředí pro udržování základních ukazatelů výkonu a plnění plánu pro jednotlivé *sprinty*. Navíc pak vytváří statistiky z měření napříč všemi *sprinty*. Cílem práce bylo zanalyzovat, navrhnout a implementovat zlepšení nástroje s cílem optimalizovat a zefektivnit administrativu.

4.1 Funkce

Pro správu projektů používá společnost dva oddělené Team Foundation servery. Současný nástroj čerpá data z obou těchto serverů a vytváří souhrnné statistiky a pohledy. Pro data ze serverů je pro zrychlení odezvy použita cache v relační databázi. Team Foundation Server nabízí API, díky kterému je možné přistupovat ze vzdálené aplikace přímo k datům o probíhajících projektech. Funkce nástroje odpovídají velice dobře agilní vývojové metodice Scrum, kterou společnost používá pro vedení projektů. Pohled aktuálního *sprintu* umožňuje rychlou odezvu v případě problémů s plněním plánu. Plánování probíhá s ohledem na časové dispozice konkrétního *sprintu* a *revize sprintů* umožňuje dlouhodobý sběr statistik pro poučení z minulých chyb. Samozřejmostí je podpora pro více týmů. Nástroj klade důraz na intuitivní ovládání a jednoduchou správu.

4.1.1 Graf výkonnosti týmu

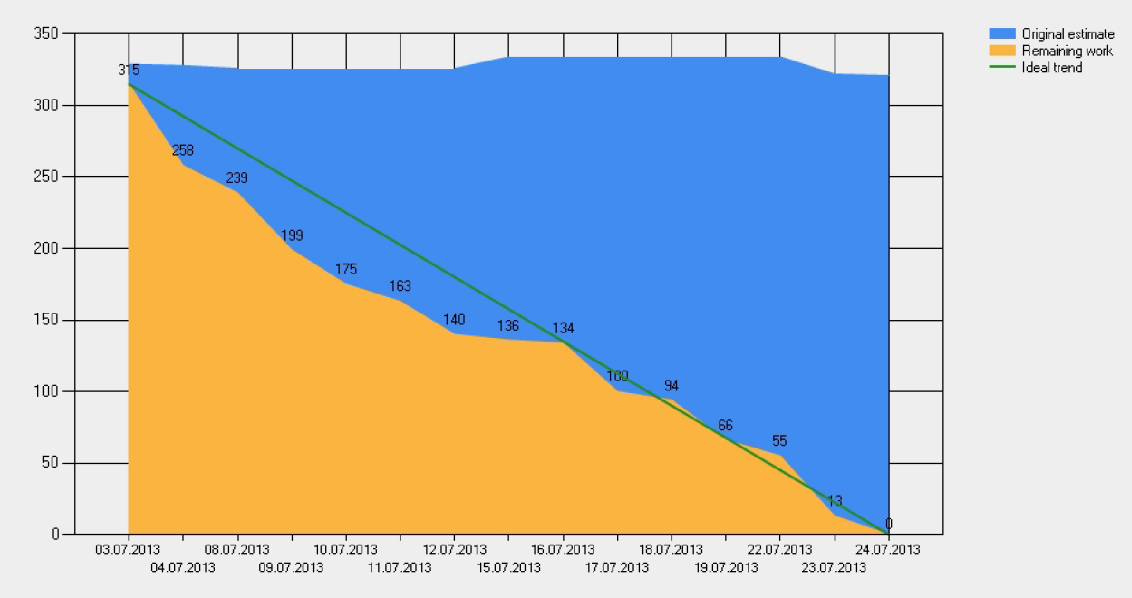
Dělení času na *sprinty* a dělení práce v katalogu úkolů umožňuje při použití metodiky Scrum detailní sledování průběhu výkonnosti týmu. Tuto skutečnost podporuje právě graf výkonnosti týmu, kinak také „Burndown chart“. Ukazuje poměr plánované a skutečně dokončené práce v průběhu času. Při správném využití tak může tým korigovat své úsilí na denní bázi. Prezentace grafu výkonnosti je vhodná např. na *denním Scrum* nebo na viditelném místě na pracovišti. V oddělení Corporate Technology se graf neustále promítá na obrazovce na pracovišti. Graf výkonnosti slouží mj. jako motivace, kdy se tým vždy snaží dosáhnout nulové zbývající práce na konci *sprintu*. Na druhou stranu, pokud nastane situace, kdy tým pracuje nad plán, může to vést k poklesu motivace a soustředění. Graf výkonnosti slouží jako jeden z hlavních vstupů do *retrospektivy sprintu*.

Z obrázku 4.1 například můžeme vyčíst téměř ideální průběh výkonnosti ve *sprintu* končící právě s nulovou zbývající prací. Hned na začátku iterace tým získal náskok vůči

ideálním trendu, který pak v průběhu *sprintu* mohl pokrýt nastalé problémy a snížení výkonnosti. Příčiny snížení výkonnosti mohou být různé:

- problémy při implementaci
- špatný odhad náročnosti
- ztráty motivace
- přepracovanost
- přesunutí pracovníků k důležitějším úkolům

Je již na týmu, jak se k interpretaci grafu postaví a jak zareaguje.

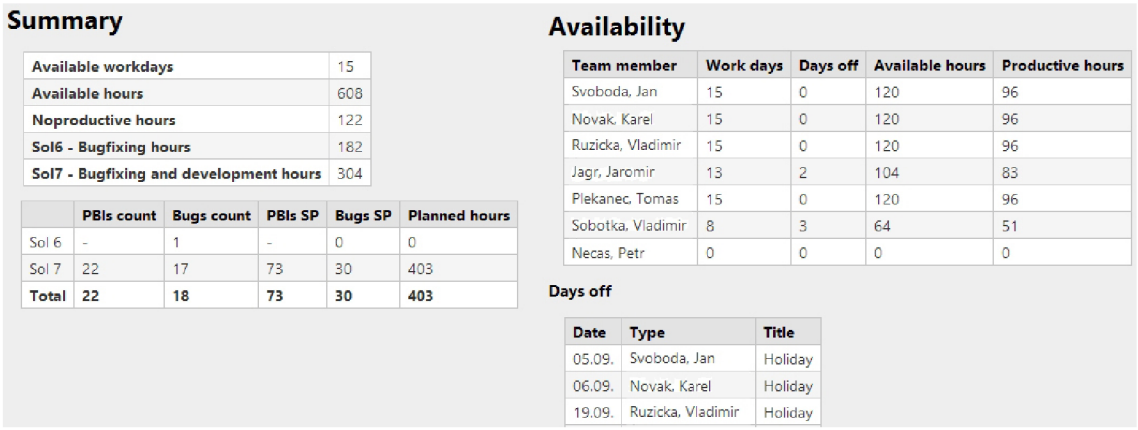


Obrázek 4.1: Graf výkonnosti týmu

4.1.2 Plánování

Funkce pro plánování především zpřístupňují časový rozvrh a dispozice jednotlivých členů týmu. Jednotlivé *sprinty* se liší v počtu pracovních hodin, které jsou ovlivněny státními svátky apod. Dále je ale do časových dispozic potřeba započítat dovolené, různá školení a služební cesty, kdy členové týmu také nejsou produktivní. Nástroj umožňuje nastavit zaměstnancům také různé úvazky. V případě ručního výpočtu by, s ohledem na všechny faktory, existovala příliš velká možnost zanesení chyby, a proto byl tento proces automatizován pomocí dříve implementovaného nástroje.

Spolu s plánováním času nabízí tato funkce přehled důvodů dovolených a textovou definici pro plánování rizik a cílů *sprintu*. Informace slouží jak pro členy týmu, tak pro management, který může regulovat školení a workshopy, případně měnit rozdělení práce pro zaměstnance.

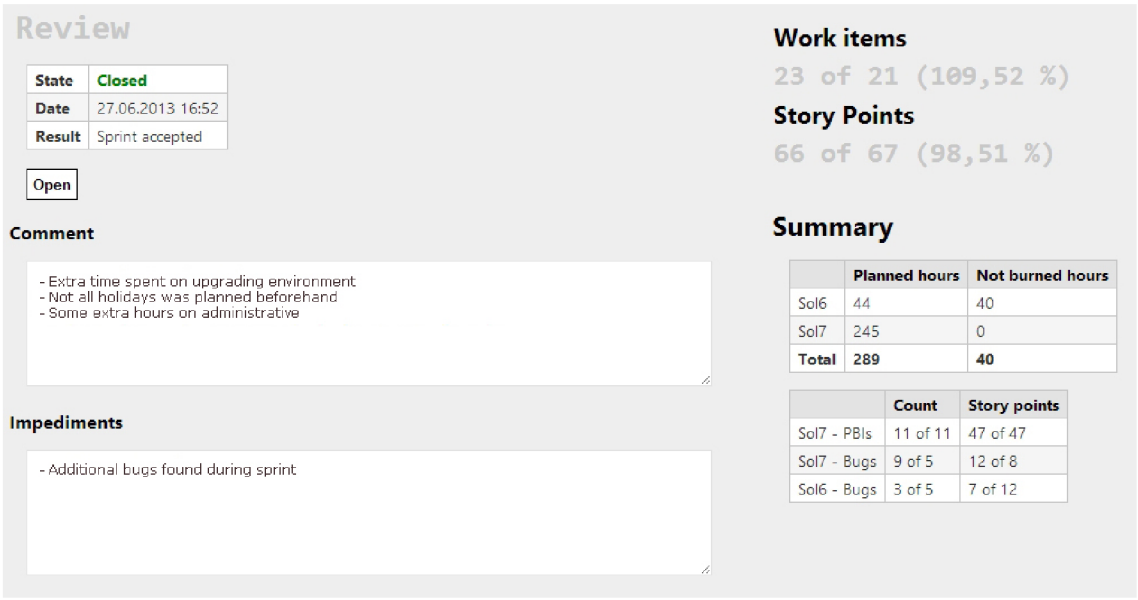


Obrázek 4.2: Obrazovka modulu Plánování

4.1.3 Revize sprintu

Po skončení *sprintu* nástroj zobrazí statistiky plánovaných a „nespálených“ hodin — rozdíl původně naplánovaných produktivních hodin a součtu odpracovaných a zbylých hodin na konci *sprintu*. Důvodem velkého množství nespálených hodin mohou být např. špatný odhad náročnosti jednotlivých úkolů nebo nemoc části týmu. Statistiky jsou dále doplněny o prostý počet splněných a nesplněných úkolů, které byly pro *sprint* určeny.

Revize dále umožňuje zadání textových komentářů k průběhu *sprintu* a překážek, které nebyly předvídány. *Revize* je dalším z prostředků pro rychlou kontrolu výkonnosti týmu. Slouží především pro samotný tým, který na jeho základě může identifikovat problémy s plánováním a průběhem *sprintů*.

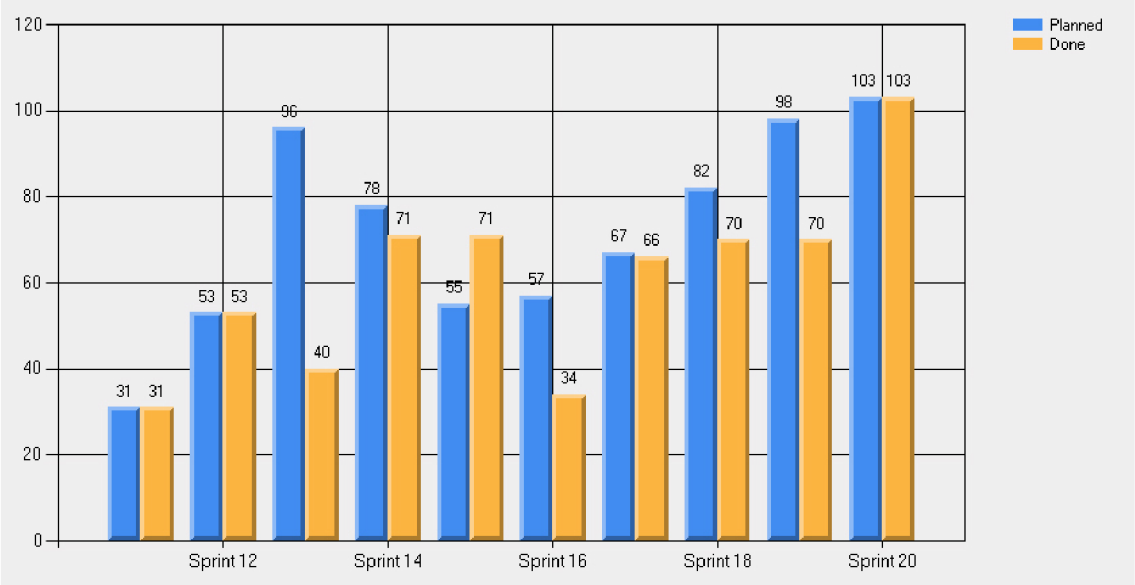


Obrázek 4.3: Obrazovka modulu Revize

4.1.4 Rychlost

Neboli *velocity* je jedinou funkcí nástroje, která není závislá pouze na jednom konkrétním *sprintu*. Udává rychlost plnění úkolů a jejím výstupem jsou dva ukazatele, které slouží především managementu ke zjištění dlouhodobé úspěšnosti týmu. Jedním z ukazatelů je průměrná výkonnost, která udává průměr celkové složitosti úkolů (pro ohodnocení složitosti slouží tzv. „story points“ splněných za poslední tři *sprinty*. Dalším ukazatelem je průměrné procento dokončených úkolů, které se počítá za veškeré *sprinty* týmu.

Ideálně by měly oba dva ukazatele výkonnosti postupně stoupat spolu s tím, jak tým získává zkušenosti a poté se ustálit na určité hodnotě. Hodnoty poté mohou vykazovat výkyvy kvůli nepředvídatelným okolnostem v průběhu projektu. Pro přehled nástroj zobrazuje také graf plánovaných a splněných úkolů v jednotlivých *sprintech*.



Obrázek 4.4: Obrazovka modulu Rychlost

4.2 Návrh nových modulů

Na základě funkcí aktuálního nástroje a Scrum technik používaných v oddělení Corporate Technology vyvstává několik oblastí pro zlepšení a automatizaci procesů, které budou popsány dále.

4.2.1 Revize kódu

Vývojový tým již nyní využívá Scrum techniky skupinové *revize kódu*. Pořádá se nezávisle na *sprintech* - někdy vícekrát, ale někdy ani jednou za *sprint*.

Revize se mohou účastnit kromě členů týmu také externisté a pozvání probíhá formou rozeslání Outlookové události. Účastníci *revize* se následně sejdou v zasedací místnosti, případně v telekonferenci. V průběhu *revize* prochází členové týmu svůj zdrojový kód a ostatní účastníci *revize* hledají chyby nebo neefektivní zápisy apod. Výsledkem *revize* je zápis chyb k opravení a také souhrn obecných ustanovení a pravidel, která by se do budoucna měla dodržovat (např. sjednocení zápisu konstant apod.).

Po skončení *revize* se nalezené chyby rozešlou autorům kódu spolu s termínem opravy. Veškerý průběh *revize* je momentálně řízen ručně a neprobíhá žádná automatizace. Zápisy se vytváří ve formě textového dokumentu na interním serveru TrackWiki, z kterého se následně vytváří pomocí Outlooku úkoly pro jednotlivé členy týmu. Stav opravy je dále sledován pouze ručně Scrum mistrem.

Možná vylepšení

Zde se tedy nabízí integrace zápisů a vedení procesu *revize* do současného nástroje ve formě nového modulu. Především by měla být zajištěna automatizace rozeslání pozvánek, rozeslání úkolů a kontroly stavu oprav. Dále obecná ustanovení z jednotlivých Revizí jsou momentálně zobrazena pouze v zápisu ze schůzek. Souhrn obecných ustanovení a zobrazení na samostatné nástěnce by mohl zvýšit jejich účinnost a přinést větší užitek pro tým.

4.2.2 Retrospektiva

Po skončení každého *sprintu* tým pořádá dle zásad metodiky Scrum tzv. *retrospektivu*. Ta spočívá ve skupinové analýze průběhu právě skončeného *sprintu*. Konkrétně se jedná o skupinové vyplnění dokumentu, který je rozdělen do osmi bodů:

Plánování - zda bylo plánování provedeno korektně a neobjevily se problémy nebo přílišné odchylky

Změny během sprintu - co se muselo během *sprintu* změnit a jaký to mělo dopad na průběh práce

Ukončení - zda bylo vše dokončeno, případně proč ne

Komunikace - zda trpěl průběh *sprintů* na nedostatek komunikace a zda měl tým všechny potřebné informace k dokončení práce

Prostředí - nedostatky firemního prostředí

Nástroje - problémy týkající se softwarového a hardwarového vybavení

Promarněné úsilí - zbytečná práce, která se musela dělat navíc

Rizika - rizika nebo příležitosti v souvislosti s průběhem *sprintu*

Výsledkem *retrospektivy* jsou také úkoly pro členy týmu, které mají zajistit, aby se problémy momentálně ukončeného *sprintu* neopakovaly. Soupis těchto úkolů společně s termíny realizace se uchovávají v dokumentu společně se zápisem z *retrospektivy*.

Možná vylepšení

Opět se zde objevuje možnost pro automatizaci rozeslání pozvánek a integraci výsledků *retrospektivy* do současného nástroje. Dále je možné automatizovat přidělování úkolů a sledování jejich stavu.

4.2.3 Týdenní setkání

V rámci *sprintů* probíhá tzv. *týdenní setkání*, kterého se účastní zákazník a vývojový tým. Účelem setkání je informování o aktuálním průběhu *sprintu* a plnění plánu. Předávají se především informace o tom, které úkoly byly dokončeny od minulého setkání, které jsou aktuálně rozpracovány a zda jsou některé ve zpoždění. Vytváření reportů k týdennímu setkání je momentálně prováděno ručně a je zapotřebí sjednotit data ze dvou Team Foundation serverů, které oddělení využívá.

Možná vylepšení

Zde se nabízí možnost velké úspory času v automatizaci vytváření těchto reportů. Zákazník i vývojový tým by tak měli veškeré informace okamžitě k dispozici. Rozdělení na dokončené, rozpracované a případně zpožděné úkoly by navíc bylo možné provázat s následujícím navrhovaným modulem - grafickou nástěnkou.

4.2.4 Grafická nástěnka

Velké množství nástrojů analyzovaných v kapitole 3 nabízelo grafickou nástěnku, která umožňovala sdílení informací o aktuálním stavu jednotlivých úkolů. Momentálně vývojový tým využívá metodiku Kanban pomocí tabule na pracovišti, a tak se nabízí možnost převedení do elektronické podoby. Výhodou elektronické nástěnky je možnost sdílení například se spolupracujícími týmy apod.

Inspiraci pro funkce nástěnky je možné hledat například na obrázku 3.2 nebo 3.7. Nástěnka by měla podporovat základní zobrazení, ve kterém se úkoly přesouvají mezi oddíly *ke zpracování*, *rozpracováno* a *hotovo*. Úkoly by měly být zobrazeny jako bloky zobrazující informace o prioritě, přiřazení apod.

4.3 Další dílčí vylepšení

Autentizace - nástroj momentálně nenabízí možnost autentizace uživatelů a je otevřený pro přístup všem členům týmu bez rozdílu dle zásad metodiky Scrum. Zavedení autentizace by ale umožnilo například logování změn a zkrácení formulářů o položku autora.

Dvousloupcové uspořádání - aktuálně se v některých modulech nachází mnoho informačních bloků s malou šířkou a velkou délkou. Bylo by tedy vhodné zavést dva sloupce a minimalizovat tak nutnost posouvání stránky například u modulů Plánování nebo Revize sprintu.

Formátování vstupů - bylo by vhodné pro vstupy použít formátování například pomocí HTML tagů pro zvýšení kvality uživatelského rozhraní. Tým používá pro většinu vstupních dat formátování do odrážek a jeho nativní podpora by přinesla větší přehlednost.

Konfigurační rozhraní - některé řídící funkce nástroje se provádí přímým přístupem do databáze, což by se mohlo v budoucnu změnit vytvořením konfiguračního rozhraní s formuláři pro správu *sprintů*, týmů, členů atd.

Nápověda - pro ulehčení nasazení nástroje v nových týmech by bylo vhodné vytvořit nápovědu a vložit poznámky přímo k jednotlivým funkcím formulářů.

Obrazovka pro promítání - na pracovišti oddělení je momentálně umístěna obrazovka, na které se promítá graf výkonnosti týmu a bylo by možné vytvořit speciální konfigurovatelnou stránku pro promítání dle aktuálních požadavků týmů.

Pohledy globální a za sprint - informace v nástroji by bylo vhodné rozdělit do dvou pohledů - globální a za sprint. Zjednodušila by se tak orientace v nástroji, kdy modul Rychlost poskytuje pouze globální data a ostatní současné moduly pouze pohledy na sprint.

Responsivní uživatelské rozhraní - několik funkcí nástroje má pomalejší odezvu, přitom se mění pouze malá část obsahu stránky a bylo by tak vhodné využít dynamického načítání. AJAX by se dal využít například v modulu Plánování pro správu dovolených.

Vizualizace dat - některá data v současném nástroji by bylo možné zpřehlednit využitím vizualizace do grafu. Jedná se především o souhrnné informace pro počty chyb a uživatelských scénářů v modulech Plánování a Revize.

Zrychlení nástroje - některé pomalejší odezvy nástroje by bylo vhodné zrychlit optimalizací přístupu do databáze. Konkrétně by se mohla optimalizovat konfigurace mapování databáze v nHibernate a minimalizovat nutnost přistupů do databáze.

Kapitola 5

Návrh a specifikace rozšíření

Následující kapitola shrnuje technologie, které budou použity pro implementaci projektu. Dále se zabývá návrhem systému dle specifikací určených firemním prostředím společnosti Siemens, s.r.o.

5.1 Volba modulů k implementaci

Na základě konzultace se zástupcem oddělení Corporate Technology byly pro implementaci zvoleny dva hlavní moduly a několik dalších dílčích vylepšení. Hlavním aspektem při výběru položek pro implementaci byl co největší přínos pro oddělení. Dalším aspektem byl potenciál pro rozšíření na další projekty společnosti. Počet položek k implementaci byl volen s ohledem na časové omezení vypracování diplomové práce, ale s možností dalšího rozšíření v průběhu implementace.

Prvním modulem zvoleným pro implementaci je *revize kódu*. Modul nabízí pro tým možnost úspory času především automatizací rozesílání pozvánek a kontrolou oprav. Kvalitu celého procesu revize kódu navíc zvýší použití globálního katalogu pro navrhování objektů k revizi a katalogu závěrů z revizí. Modul navíc nabízí velký potenciál pro rozšíření do dalších oddělení společnosti, které budou zavádět proces revize kódu.

Modul Retrospektiva sprintu byl zvolen pro další úsporu času a zvýšení kvality procesů. Jeho specifikace jsou navíc podobné modulu *revize kódu* a při implementaci tak bude možné znovupoužití některých funkcí. Stejně jako u předchozího modulu tak budou použity funkce spojení s poštovním klientem Outlook a globální katalog pro závěry.

Z dílčích vylepšení byly zvoleny:

Dvousloupcové uspořádání u všech modulů

Formátování vstupů pomocí HTML u všech modulů

Pohledy globální a za sprint v celém nástroji

Responsivní uživatelské rozhraní v modulu Plánování pro správu dovolených

Vizualizace dat u modulů Plánování a Revize pro souhrnná data

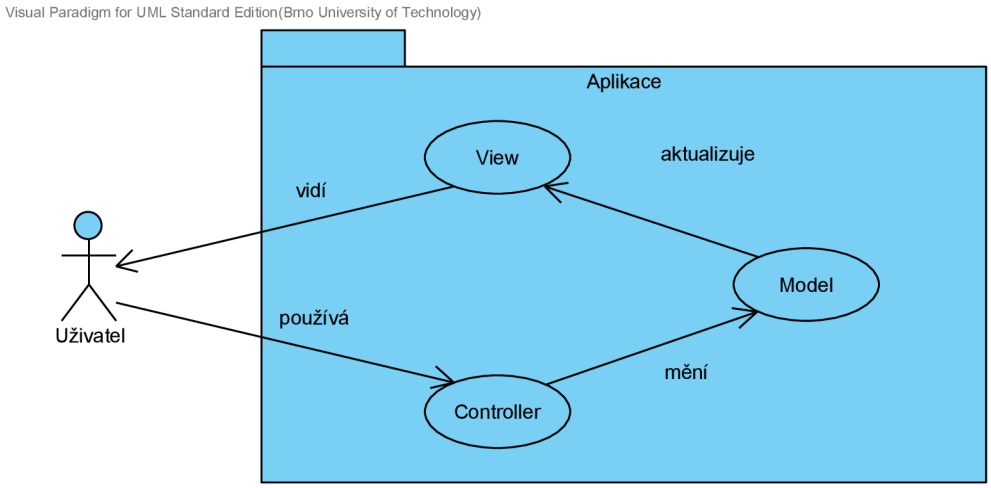
5.2 Prostředky

Většina technologií je určena současnou verzí nástroje a nastaveným firemním prostředím a nebylo tedy zapotřebí provádět výběrové řízení.

5.2.1 MVC4

MVC4 je framework postavený na ASP.NET určený pro webové aplikace. Aplikuje v internetových aplikacích hojně využívaný návrhový vzor model-view-controller. V současné době již existuje ve verzi 5, ale kvůli spolupráci s původním nástrojem bude dále využita verze 4.

Pro MVC framework je určeno vývojové prostředí společnosti Microsoft – Visual Studio. V současné době je již k dispozici aktuální verze 2013, nicméně vývoj bude probíhat ve verzi 2012 především kvůli dostupnosti a kompatibilitě knihoven s původní verzí nástroje.



Obrázek 5.1: Podstata návrhového vzoru MVC

Návrhový vzor MVC rozděluje aplikaci do tří částí. Pro interakci s aplikací využívá uživatel řídící logiku umístěnou v controlleru. Controller zajišťuje korektní zpracování požadavků a změny v datovém modelu. Datový model se udržuje v části model, ke kterému uživatel nemá přímý přístup. Při změně v datovém modelu pak dochází k aktualizaci uživatelského rozhraní. To je umístěno v části view a zajišťuje transformaci dat z modelu do podoby vhodné pro zobrazení uživateli. Principy se mohou v různých implementacích návrhové vzoru mírně lišit, avšak v tomto projektu bude důsledně dodržována standardní forma.

Při vývoji se používá kombinace kódu v jazyce C#, XML a dále speciálního jazyka Razor pro vytváření uživatelských pohledů. Razor doplňuje klasické webové technologie XHTML, JavaScript a CSS o syntaxi pro dynamickou tvorbu kódu a jeho navázání na řídící logiku i datový model.

5.2.2 nHibernate

nHibernate je řešení objektově-orientovaného mapování relační databáze pro platformu .NET. Objektově-orientované mapování přináší pro vývoj aplikací mnoho výhod, z nichž hlavní je oproštění od programování v jazyce SQL a usnadnění práce s persistentně uloženými daty v databázi. Díky tomu se mohou programátoři při vývoji soustředit pouze na objektový model a programování v jednom jazyce.

Pomocí knihovny nHibernate vytvoří vývojář mapování, které po prvním spuštění kódu automaticky zajistí vytvoření příslušných MSSQL tabulek včetně klíčů a integračních a datových omezení. Následně pracuje vývojář pouze s objekty reprezentujícími data, ale kon-

krétní funkce pro práci s relační databází zůstávají zapouzdřena. Knihovna je poskytována pod volnou licencí MIT.

Ukázka mapování a vytvoření objektu:

```
// mapování databáze
public class CodeReviewMap : ClassMap<CodeReview>
{
    public CodeReviewMap()
    {
        Id(x => x.Id);
        Map(x => x.Time);
        Map(x => x.State).CustomType(typeof(StateEnum));

        References(x => x.Sprint).Column(,,SprintId‘‘);
        References(x => x.Team).Column(,,TeamId‘‘);

        HasManyToMany(x => x.TeamMembers).Cascade.None();
        HasMany(x => x.ProposedInventory).Cascade.Delete();

        Table(,,CodeReview‘‘);
    }
}

// definice objektu
public class CodeReview
{
    public virtual int Id { get; set; }
    public virtual DateTime Time { get; set; }

    public virtual Sprint Sprint { get; set; }
    public virtual Team Team { get; set; }
    public virtual StateEnum State { get; set; }

    public virtual IList<TeamMember> TeamMembers { get; set; }
    public virtual IList<ProposedInventory> ProposedInventory { get; set; }
}
```

5.2.3 jQuery

Pro interaktivní uživatelské rozhraní bude využito javascriptového frameworku jQuery. Framework nabízí připravené rozhraní pro javascriptové asynchronní volání, tedy moderní technologii dnes známou jako AJAX.

Technologie AJAX umožňuje dynamickou změnu uživatelského pohledu bez nutnosti aktualizovat stránku. Zrychluje tak práci s uživatelským rozhraním a umožňuje použití technik, které jsou jinak možné pouze v desktopových aplikacích. Vzhledem k přínosu technologie AJAX je jedním z cílů implementace také aplikace technologie a principů dynamické změny uživatelského pohledu do původní verze nástroje.

Důležitou součástí frameworku jQuery je zpřístupnění selektorů používaných standardně pouze v CSS. Díky tomu jsme schopni efektivnějšího zápisu kódu a výrazného zvýšení míry

znovupoužitelnosti kódu. Použití selektorů navíc podporuje princip oddělení javascriptu a HTML, který umožňuje vytváření přehledného, strukturovaného kódu.

Aktuálně je jQuery oficiálně integrováno také do platforem společnosti Microsoft (tedy i MVC). Pro jQuery existuje mnoho rozšiřujících zásuvných modulů, z nichž v projektu bude využito jQuery UI pro pozvednutí úrovně uživatelského rozhraní. Zásuvný modul nabízí podporu dialogových oken, formuláře pro výběr data, zobrazení popisků při najetí myší apod. V neposlední řadě také sjednocuje ikony uživatelského rozhraní. jQuery je poskytováno pod volnou licencí MIT, stejně tak všechny jeho knihovny a rozšíření.

5.2.4 Microsoft Outlook

Pro týmovou komunikaci a spolupráci bude využit poštovní klient Outlook, který je momentálně ve společnosti Siemens, s.r.o. používán ve verzi 2007. Poštovní klient zajišťuje kromě standardní správy pošty také udržování kalendáře úkolů a schůzek. Tyto jsou využívány pro synchronizaci týmu. Nástavba aplikace pro komunikaci se serverem Microsoft Lync navíc umožňuje pořádat živé schůzky s podporou přenášení obrazu i zvuku.

Outlook nabízí API, díky kterému je možné komunikovat přímo s aplikací napsanou v jazyce C#. API umožňuje programově spravovat schůzky, ale také úkoly a tyto synchronizovat například s relační databází aplikace. Společnost Microsoft udržuje knihovny pro práci s API konzistentní od Outlook verze 2003, čímž je zajištěna kompatibilita při upgrade programového vybavení společnosti.

5.3 Návrh

Hlavní částí rozšíření nástroje bude modul pro pravidelnou revizi kódu. Tento modul musí spravovat kompletní proces od navrhování částí kódu k revizi, přes uspořádání samotné revize až po rozdělení a správu oprav kódu.

5.3.1 Specifikace požadavků

Pro rychlou specifikaci modulů Revize kódu a Retrospektiva sprintu budou využity use case diagramy dle definice UML.

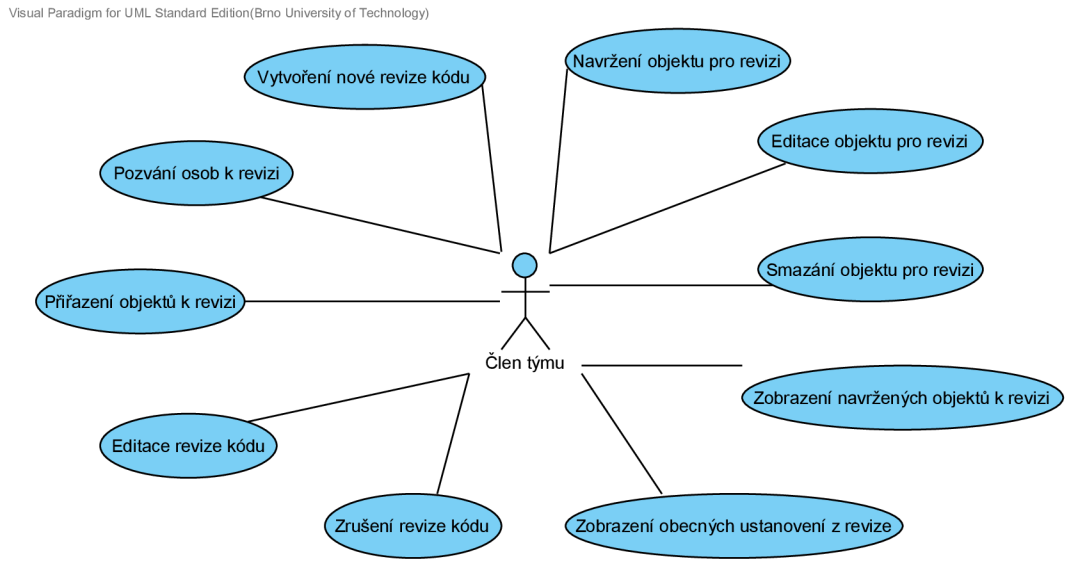
Plánování revize kódu

Ve fázi plánování revize kódu mají všichni členové týmu možnost navrhnout části kódu k revizi. Návrh musí obsahovat komentář včetně označení souborů, které budou ovlivněny a budou potřeba přezkoumat. Návrh musí dále uchovávat autora a čas vytvoření. Návrhy musí být kompletně editovatelné a dohledatelné.

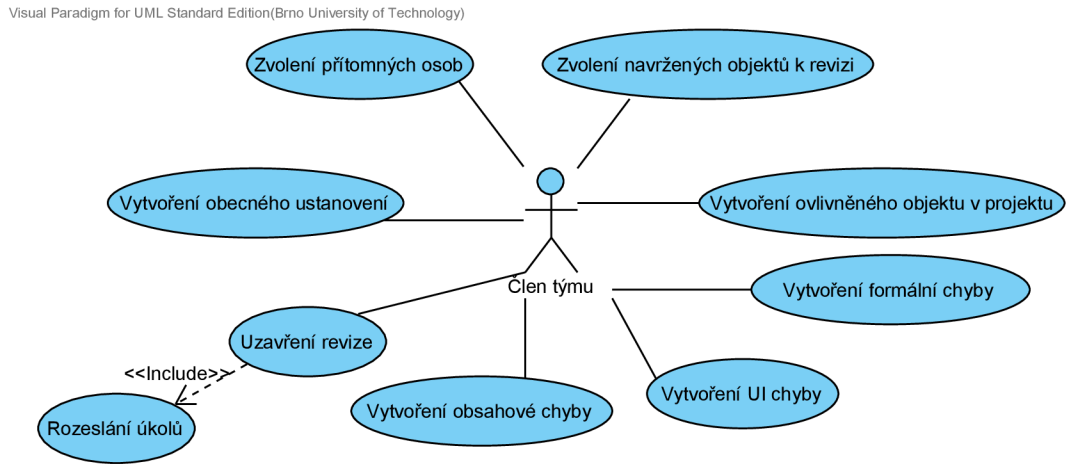
Revize kódu se rozlišují pořadovým číslem a rokem. Naplánování nové revize vyžaduje určení *sprintu*, ke kterému se revize vztahuje, dále pozvání určených členů týmu pomocí kalendáře Outlook, označení navržených objektů k revizi a zadání data, času a délky trvání. Všechna tato data revize musí být editovatelná včetně úpravy data a času konání v kalendáři pozvaných členů.

Průběh revize kódu

Po zahájení revize kódu je nutné zrevidovat všechny přítomné členy týmu – ať už odmítli pozvánku na setkání přes Outlook nebo se nedostavili bez omluvy. Po zahájení jsou pak



Obrázek 5.2: Use case pro fázi před revizí kódu



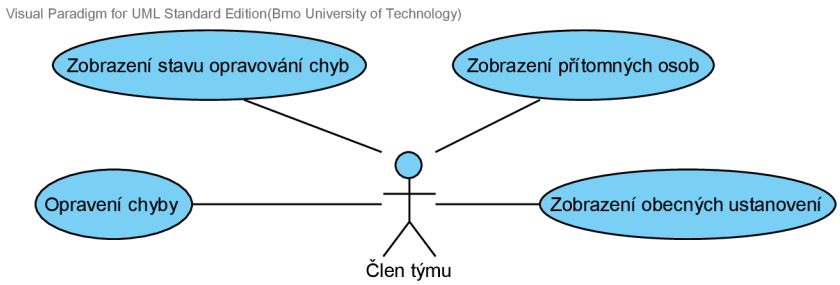
Obrázek 5.3: Use case pro průběh revize kódu

vybrány navržené problémy k revizi dle schopností přítomných a časových možností (revize musí mít určenou délku). K jednotlivým problémům jsou přiřazeny ovlivněné soubory v projektu – aplikace musí umožňovat také přidání nových a další možnosti editace souborů.

Jednotlivé ovlivněné soubory jsou postupně revidovány týmem, v průběhu čehož jsou zapisovány chyby. Existují tři různé druhy chyb – obsahová, formální nebo chyba uživatelského rozhraní. Každý soubor se přiřazuje jedné osobě, která poté zajistí opravu. V průběhu revize kódu je dále možno narazit na problém, kterému by se do budoucna měli členové týmu vyhýbat – takový problém se zformuluje do obecného ustanovení. *Revize kódu* skončí uplynutím času schůzky, případně vyčerpáním objektů k revizi.

Po skončení revize dostane Scrum mistr možnost zrevidovat vzniklé úkoly a jejich přiřazení členům týmu. Po této revizi jsou pak úkoly automaticky rozeslány do Outlook kalendářů jednotlivých členů zodpovědných za opravy chyb v kódu.

Fáze po skončení revize

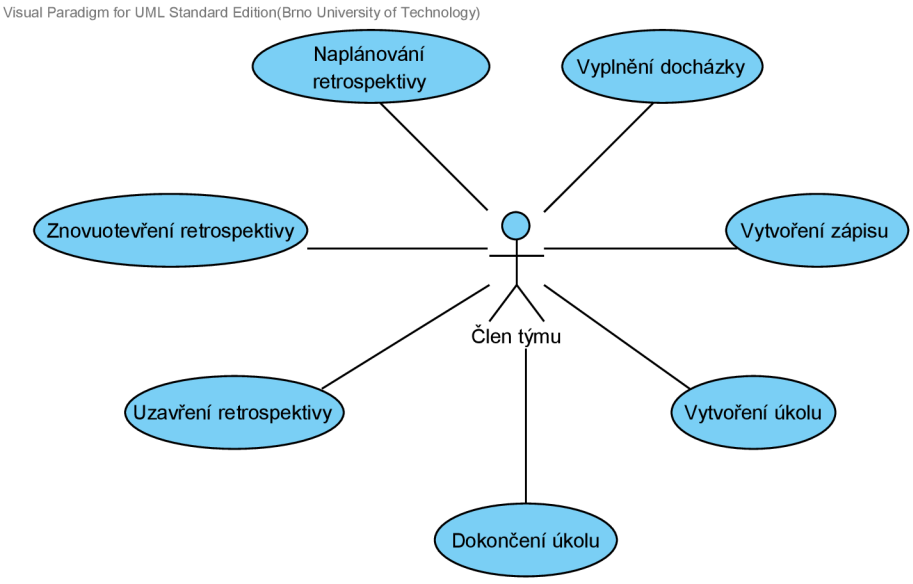


Obrázek 5.4: Use case pro fázi po skončení revize

Po skončení revize mají všichni členové týmu možnost vidět nalezené chyby a jejich stav opravení. Pokud zodpovědný člen označí svůj úkol opravy kódu v Outlooku za hotový, pak se automaticky aktualizuje stav úkolu v modulu. Všichni členové týmu tak vidí aktualizovaný stav oprav.

Obecná ustanovení se členům týmu zobrazují nezávisle na aktuálním *sprintu* nebo aktuálně vybrané revizi kódu, jelikož se jedná o ustanovení, která by měla vést k zvýšení efektivity týmu (především pak neopakovat stejné chyby dvakrát). Zobrazení by mělo být pro přehlednost k dispozici na samostatné nástěnce obecných ustanovení.

Retrospektiva sprintů



Obrázek 5.5: Use case modulu pro správu retrospektiv

Modul retrospektivy sprintů umožňuje každému členu týmu naplánovat retrospektivu a rozeslat pozvánku. Samotné vyplňování zápisu začíná vyplněním docházky. Po tomto kroku dostanou uživatelé možnost vyplnit zápis k jednotlivým aspektům sprintu a dále vytvořit úkoly. Úkoly z minulých retrospektiv je při vytváření zápisu možné označit jako dokončené. Po vyplnění retrospektivy je možné zápis uzavřít, případně poté znovu otevřít.

5.3.2 Schéma Databáze

Pro zjednodušení diagramů jsou databázové tabulky původního nástroje znázorněny pouze entitou bez atributů.

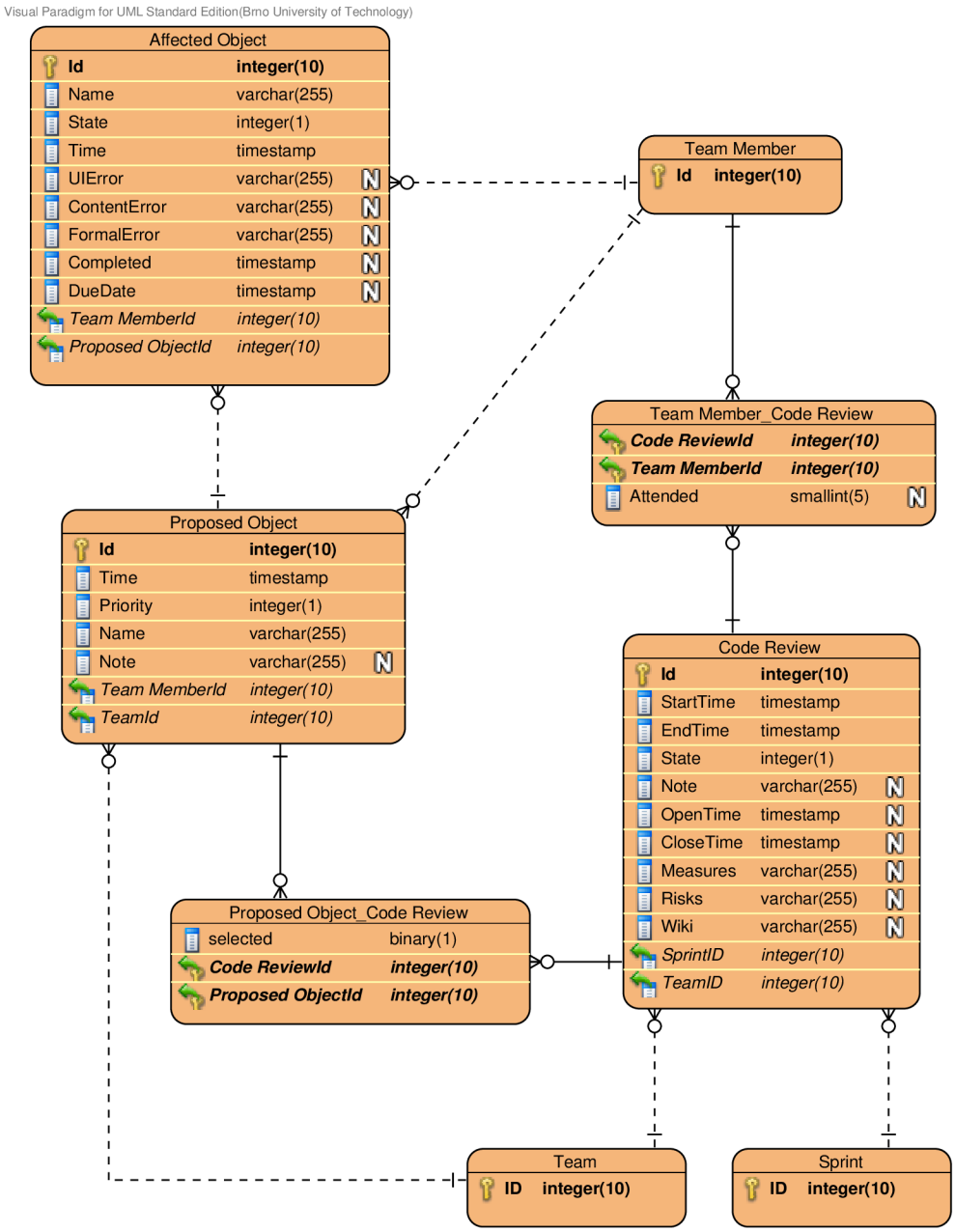
Modul Revize kódu, jehož schéma databáze se nachází na ER diagramu 5.6, vyžaduje tři hlavní entity - samotnou revizi, navrhovaný objekt a ovlivněný objekt. Entita revize uchovává informace o plánovaném začátku a konci, jakož i o skutečném začátku a konci určeném dle vytváření zápisu. Dále obsahuje informaci o stavu, poznámku pro zúčastněné, obecná ustanovení, rizika a v případě, že se jedná o historickou revizi, tak také její URL. *Revize kódu* je přiřazena k týmu a konkrétnímu sprintu a dále obsahuje vazbu na pozvané účastníky s příznakem, zda se skutečně dostavili.

Další entitou je navrhovaný objekt, který obsahuje informaci o názvu, popisu a času vytvoření. Navrhový objekt má dále vazbu na autora a revizi kódu. Vazba s revizí kódu se navíc může nacházet ve dvou stavech a sice zvolená a nezvolená.

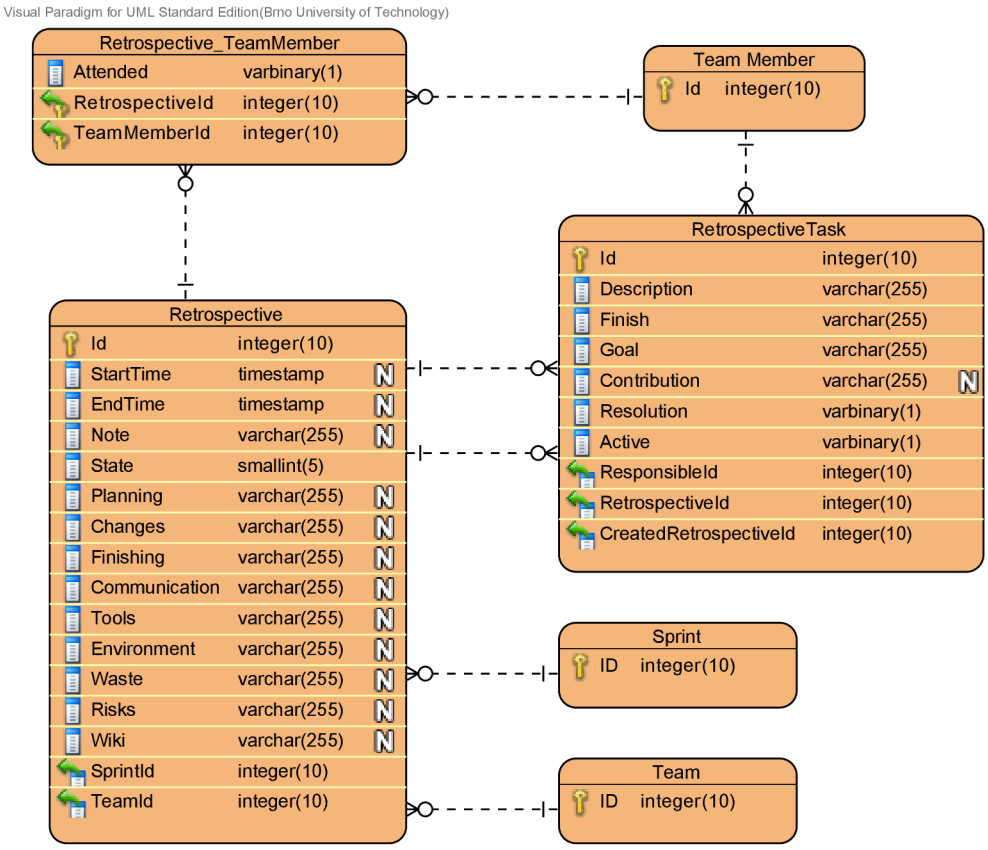
Poslední entitou je ovlivněný objekt, který uchovává informace o konkrétních nalezených chybách v kódu. Ty jsou uloženy jako řetězec ve třech částech - formální, obsahové a chyby uživatelského rozhraní. Ovlivněný objekt dále obsahuje informaci o názvu (třída, soubor apod.), stavu, času vytvoření a data, do kterého mají být nalezené chyby opraveny. Entita ovlivněného objektu se přímo váže na navrhovaný objekt, ke kterému spadá a dále na člena týmu, který je zodpovědný za opravu nalezených chyb.

Modul Retrospektiva sprintu uchovává data pouze ve dvou hlavních entitách zobrazených na ER diagramu 5.7. Jedná se o entitu pro samotnou retrospektivu, která udržuje informace o stavu, popisu a plánováném začátku a konci retrospektivy. Dále pak uchovává v řetězcích zápisy k jednotlivým aspektům sprintu - konkrétně se jedná o plánování, změny, dokončování, komunikaci, nástroje, prostředí, plýtvání a rizika. Posledním uchovávánou informací je pak odkaz v případě historické retrospektivy. Entita je vázána na pozvané členy týmu společně s příznakem, zda se skutečně zúčastnili retrospektivy. Dále se zápis retrospektivy váže na konkrétní tým a sprint.

Druhou entitou je pak úkol. Ten sestává z popisu, cíle, přínosu, slovního popisu předpokládaného data dokončení a příznaků aktivity a dokončení. Entita je vázána na zodpovědné osoby a dále na retrospektivu, ve které úkol vznikl a retrospektivu, do které je úkol přiřazen.



Obrázek 5.6: Schéma databáze modulu Revize kódu



Obrázek 5.7: Schéma databáze modulu Retrospektiva sprintu

Kapitola 6

Implementace a testování rozšíření

Tato kapitola se zabývá popisem implementovaného řešení obou hlavních modulů i dílčích změn současného nástroje. Popis je dále doplněn ukázkami uživatelského rozhraní.

6.1 Modul Revize kódu

Tento modul je stěžejní částí diplomové práce a s druhým implementovaným modulem Retrospektiva sprintů má mnoho společných prvků. Modul má dvě hlavní obrazovky - přehledovou a pro konkrétní revizi. Přehledová obrazovka je dále rozdělena na dva různé pohledy a sice globální a za sprint. Obrazovka pro konkrétní revizi slouží nejprve k zadávání dat v průběhu revize a dále pro kontrolu a přehled opravených chyb.

6.1.1 Přehledový pohled

Globální pohled nabízí data z veškerých revizí kódu přidružených k aktuálně zvolenému týmu. Pohled za sprint (ukázka na obrázku 6.1) filtruje pouze data související s jedním konkrétním sprintem. Pohledy jsou vhodné pro případný audit, ale také pro zpřehlednění informací pro samotný tým.

Údaje jsou zobrazeny textově ve formě výpisu všech revizí kódu, výpisu navržených objektů k revizi a výpisu obecných ustanovení a rizik z proběhlých revizí. Dále jsou data zpracována také vizuálně do grafů, které znázorňují statistiky délky revizí, počtu nalezených chyb apod. Následuje popis všech částí přehledového pohledu mimo statistik, kterým se věnuje samostatná kapitola 6.1.3.

První sekcí přehledového pohledu je výpis všech revizí kódu a jejich stavů (ukázka na obrázku 6.2). Existuje pět různých stavů revizí kódu:

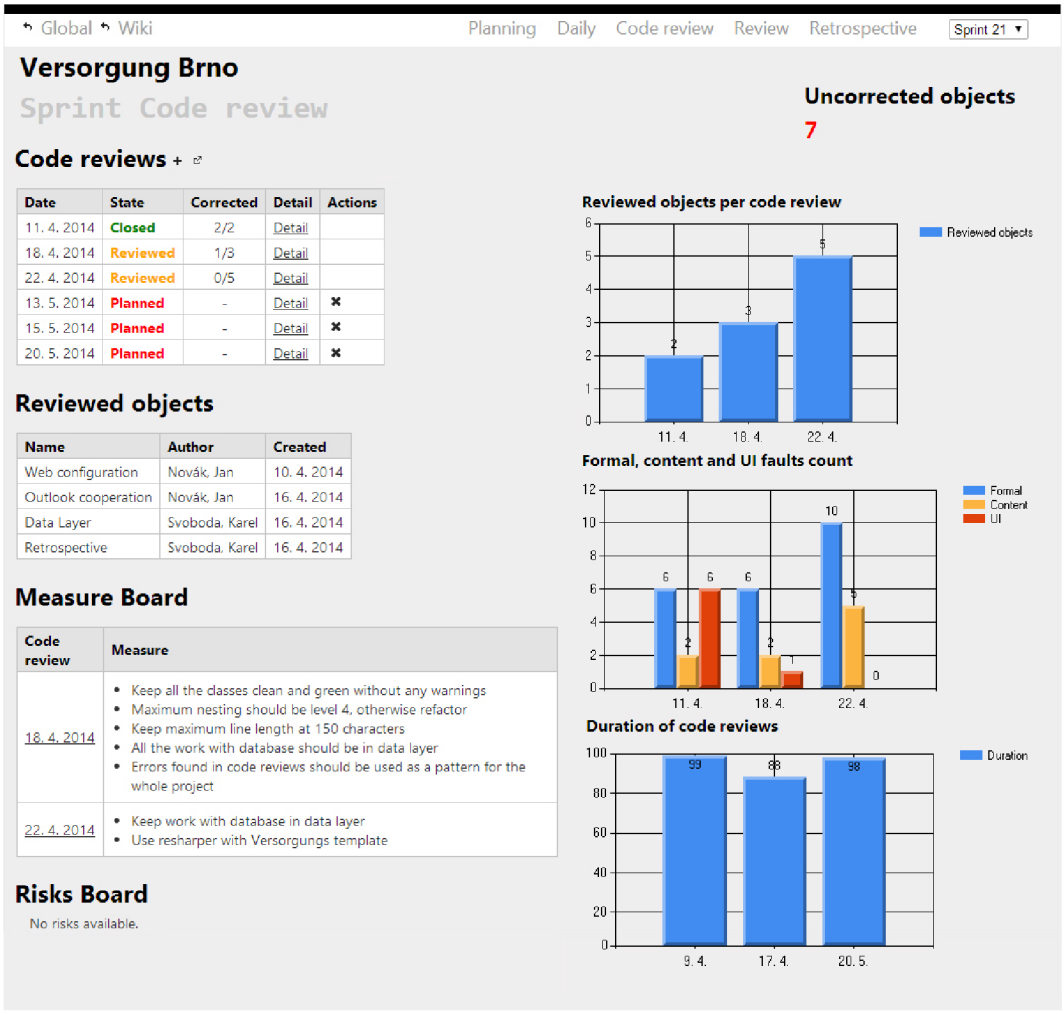
Naplánovaná - v tomto stavu se nachází revize po vytvoření a značí, že byla zvoleným členům zaslána pozvánka na událost

Otevřená - do tohoto stavu se revize přepne po vyplnění docházky a samotném začátku události

Revidovaná - stav po uzavření revize, kdy se rozešlou požadavky na opravy zodpovědným osobám

Uzavřená - slouží pro označení revize, která již má všechny opravy dokončeny

Wiki - speciální stav pro historickou revizi obsahující pouze odkaz na wiki oddělení



Obrázek 6.1: Snímek obrazovky sprint pohledu na revizi kódu

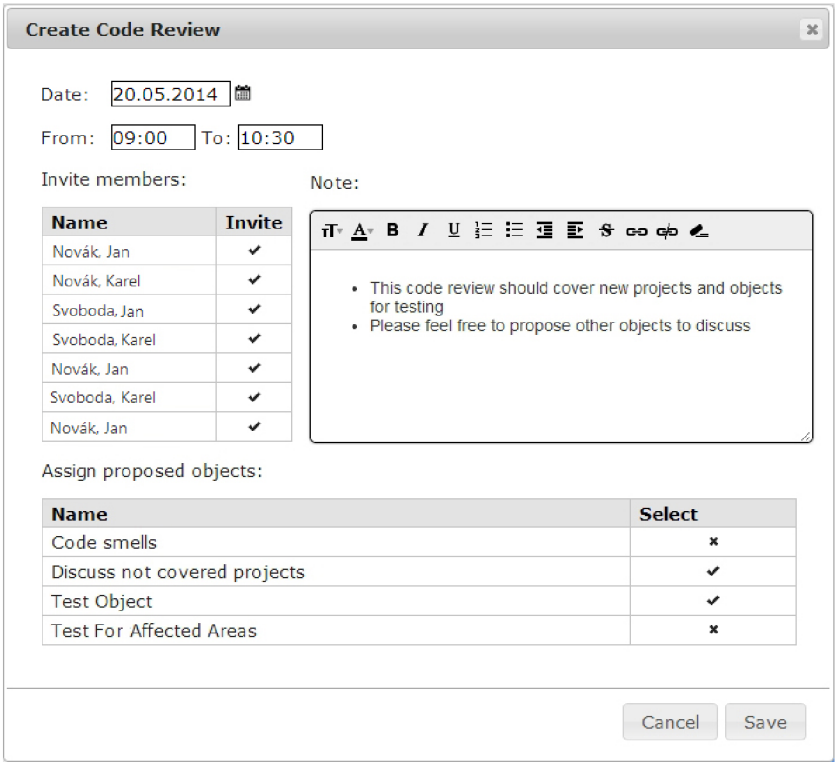
Code reviews

Date	State	Corrected	Detail	Actions
1. 4. 2014	Wiki	-	Wiki	✖
9. 4. 2014	Closed	2/2	Detail	
11. 4. 2014	Closed	2/2	Detail	
17. 4. 2014	Reviewed	0/3	Detail	
18. 4. 2014	Reviewed	2/3	Detail	
22. 4. 2014	Closed	1/1	Detail	
30. 4. 2014	Opened	-	Detail	✖
4. 5. 2014	Planned	-	Detail	✖

Obrázek 6.2: Výpis všech revizí kódu

Revize ve stavu *revidovaná* nebo *uzavřená* obsahuje informaci o celkovém počtu revidovaných objektů a o počtu neopravených objektů. Pokud se mezi objekty nachází některý již

po datu, do kterého měly být opravy dokončeny, pak je tato informace zdůrazněna barevným odlišením a s titulkem informujícím o nutné dodatečné pozornosti. U revizí ve stavu *plánovaná*, *otevřená* nebo *wiki* má uživatel možnost smazání revize.



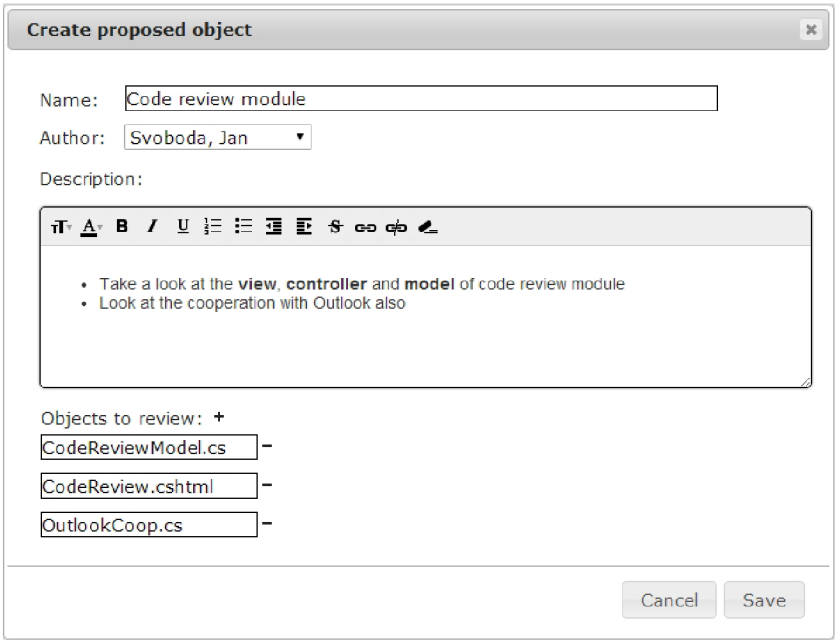
Obrázek 6.3: Formulář tvorby revize kódu

Formulář vytváření revize kódu (ukázka na obrázku 6.3) je přístupný pouze z přehledového pohledu za sprint, aby bylo jasné, ke kterému sprintu náleží. Veškeré formuláře v nástroji využívají implementaci modálních dialogů v rozšíření jQueryUI frameworku jQuery. Revizi může vytvářet kterýkoliv člen týmu bez rozdílu. První část formuláře pro vytváření revize tvoří zadání data a času. Pro zadávání data a času v celém nástroji slouží grafické prvky kalendáře a posuvníků vytvořené také pomocí rozšíření jQueryUI.

Další částí formuláře je zadání popisu revize. Pro vstup slouží textové pole obohacené o HTML formátování včetně nástrojové lišty. Toto formátování používá celý nástroj a staví na rozšíření jQuery TE frameworku jQuery. Poslední částí formuláře jsou pak výběry členů týmu a objektů navržených k revidování. Tyto prvky fungují jako klasické selektivní seznamy, ale mají upraveno grafické rozhraní. Po odeslání formuláře se zanese revize do databáze a vytvoří se událost v poštovním klientu.

Výpis jednotlivých navržených objektů (můžete vidět na obrázku 6.1) se v jednotlivých pohledech liší. U pohledu za sprint se vypisují všechny navržené objekty, které byly nakonec v nějaké z revizí v tomto sprintu zvoleny. U globálního pohledu se pak vypisují všechny objekty, které ještě nebyly v žádné revizi zvoleny. Rozhodování o nutnosti uspořádat novou revizi by se tedy mělo řídit globálním pohledem - jakmile se objeví dostatek objektů, je třeba vytvořit revizi.

Pro vytváření navrhovaných objektů slouží opět modální dialog (ukázka na obrázku 6.4). Navrhovaný objekt má vždy svůj název a autora. Dále se zadává volitelně popis a seznam



Obrázek 6.4: Formulář pro návrh objektů k revizi

ovlivněných objektů, tedy názvů tříd, souborů apod. Objekty k revizi lze případně přidávat až v průběhu samotné revize a tak není podmínkou jejich zadání už při navržení objektu.

Poslední částí přehledového pohledu je sekce s výpisem obecných ustanovení a rizik. Tato sekce vybírá data dle aktuálního pohledu - pro sprint nebo celkově pro tým. Data se vypisují dle HTML formátování tak, jak byla uložena v průběhu sprintu.

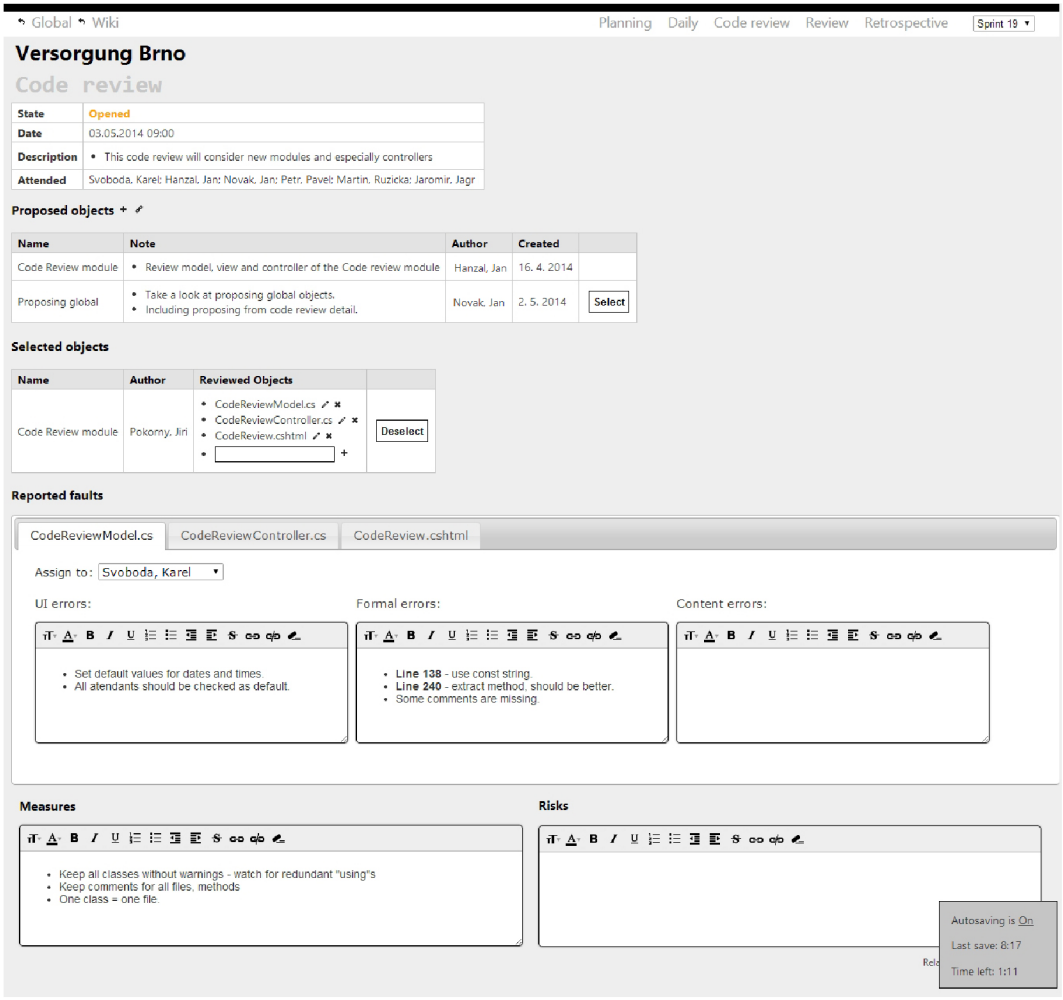
6.1.2 Průběh revize

Životní cyklus revize obsahuje několik fází. V první fázi se doplňují a vytvářejí objekty k revizi a následně se vyplní docházka. Po vyplnění docházky začíná samostatná událost revize, kdy se volí navržené objekty a vytváří se zápis. Po uzavření revize následně obrazovka nabízí přehled nalezených chyb a jejich aktuální stav opravy. Po opravení všech chyb se zobrazí pouze přehled chyb a statistiky oprav.

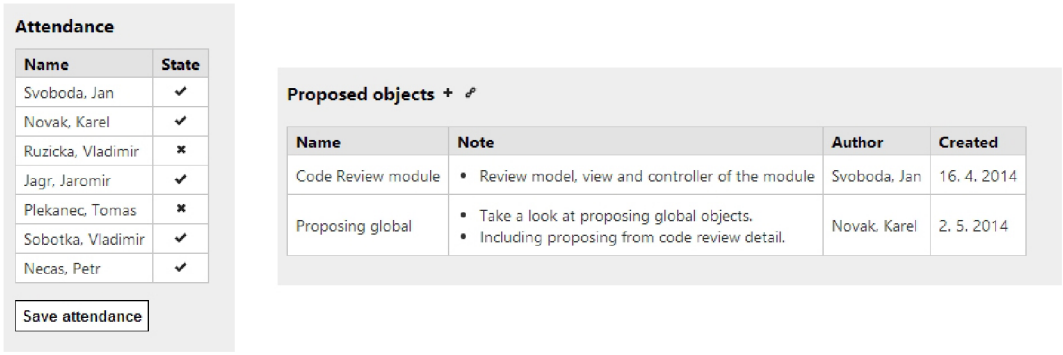
Vyplnění docházky probíhá pomocí selektivního seznamu (ukázka na obrázku 6.6). Seznam se omezuje na členy týmu, kteří byli k revizi pozváni. Docházka je předvyplněna dle reakcí členů týmu na pozvánku v poštovním klientu. V momentě odeslání účasti se začíná měřit samotný čas revize kódu pro účely vytváření statistik a dodržování stanovené délky události.

Dalším prvkem přístupným pro revize kódu ve stavu *plánovaná* je přiřazování a vytváření objektů k revizi — na obrázku 6.6. Na začátku má revize přiřazeny pouze objekty, které se navázaly přímo při vytváření události. Zde má uživatel možnost přímo vytvářet nové objekty k revizi, případně také navázat ty, které už v katalogu objektů jsou. Je vhodné tento krok provést dříve než událost začne tak, aby tým již dále neztrácel čas určený pro kontrolu kódu.

Po odeslání docházky nastává samotná fáze zapisování nalezených chyb v kódu. Tým dle časových dispozic a vlastních priorit postupně volí navržené objekty (uživatelské rozhraní na obrázku 6.7). Uživateli se zobrazí tabulka s navrženými objekty a z této postupně objekty



Obrázek 6.5: Obrazovka průběhu revize kódu

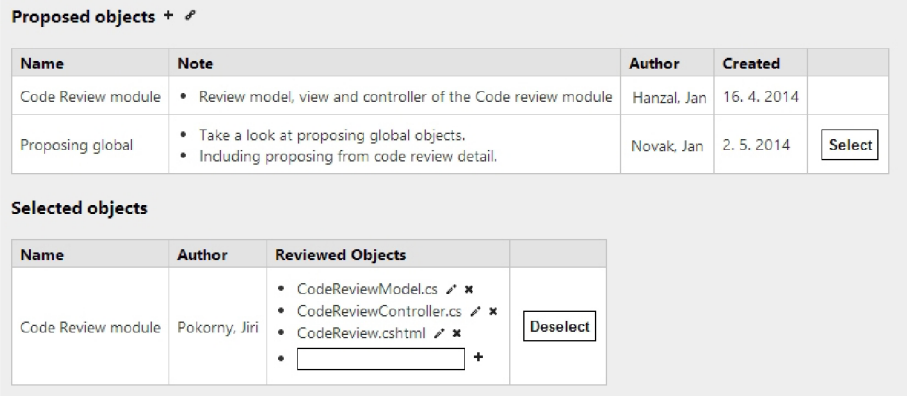


Obrázek 6.6: Snímek obrazovky vyplňování docházky a přiřazování objektů k revizi

přesouvá do tabulky se zvolenými objekty. Tabulka s navrženými objekty podává informace o názvu, popisu, datu vytvoření a autorovi. Za těmito informacemi následuje možnost pro zvolení objektu.

Po vybrání se navrhovaný objekt přesune do tabulky zvolených objektů. Zde se vypisují informace o názvu, autorovi a ovlivněných objektech spolu s možností opětovného zrušení

příznaku zvolení. Stěžejní částí této tabulky je správa konkrétních ovlivněných objektů. Jedná se tedy o názvy souborů, případně tříd nebo prvků uživatelského rozhraní. Ovlivněné objekty jsou identifikovány pouze svým názvem a tým si tedy může zvolit, která možnost mu nejvíce vyhovuje. Tyto objekty je možné přímo v tabulce libovolně přidávat, editovat nebo mazat. Pro urychlení práce uživatelské rozhraní reaguje na stisknuté klávesy a je tak možné provést proces zadávání ovlivněných objektů pouze za použití klávesnice.



Obrázek 6.7: Obrazovka volby objektů k revizi

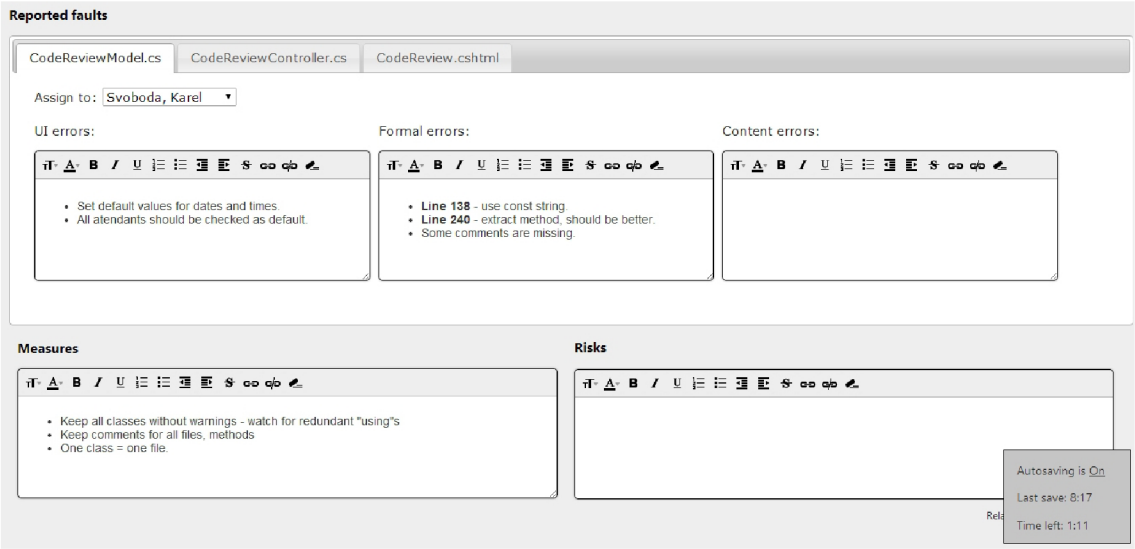
Další sekcí průběhu revize jsou formuláře pro výpis chyb, obecných ustanovení a identifikovaných rizik (ukázka na obrázku 6.8). Textové vstupy využívají upravený plugin jQuery TE pro HTML formátování. Pro umožnění vytváření statistik používají formuláře formátování vstupu do odrážek, které přímo vynucují od uživatele.

Zapisování chyb je uspořádáno do záložek. Pro každý ovlivněný objekt z tabulky zvolených objektů se automaticky vytvoří záložka. V té jsou celkem čtyři formulářové prvky — jeden selektor a tři textová pole. Selektor slouží pro zvolení osoby, která bude zodpovědná za opravu nalezených chyb v ovlivněném objektu. Textové pole pak slouží postupně pro zadávání chyb uživatelského rozhraní, formálních a obsahových. Toto rozdělení chyb vychází z dělení již zavedeného na oddělení Corporate Technology.

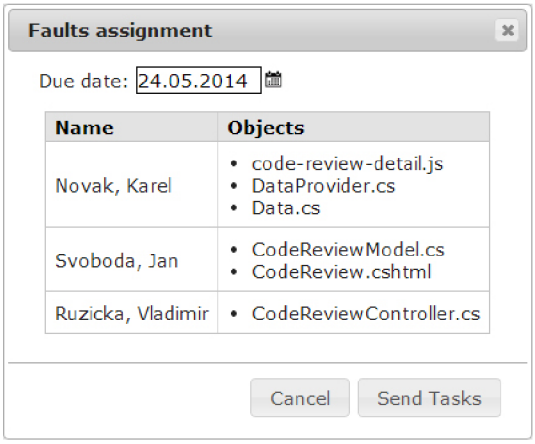
Na obrázku 6.8 je v pravém dolním rohu ukázka ovládacího panelu pro ukládání dat a odpočet zbývajícího času události. Data zadávaná do textových polí se ukládají buď po stisknutí tlačítka nebo pomocí automatického ukládání. Tuto volbu lze zapnout ve fixně umístěném ovládacím panelu, kde je zobrazena také informace o posledním uložení. Interval pro automatické ukládání je nastaven na jednu minutu a měl by dostačovat k minimalizaci rizika ztráty dat. Případná konfigurace intervalu je možná úpravou konstanty v příslušném javascriptovém souboru. Na ovládacím panelu se nachází dále odpočet zbývajícího času události dle času nastaveného při rozesílání pozvánky na revizi kódu. Deset minut před vypršením času se odpočet barevně zvýrazní a po skončení odpočtu se zobrazí dialog s upozorněním.

Před uzavřením revize kódu se zobrazí formulář pro kontrolu rozesílaných úkolů. Dále slouží také k zadání data, do kterého mají být objekty opraveny. Pokud se tak nestane, pak jsou úkoly v přehledovém pohledu barevně zvýrazněny. Ukázku formuláře je možné vidět na obrázku 6.9.

Na obrázku 6.10 vidíme ukázku již dokončené revize kódu. Nejdříve se zobrazuje hlavní tabulka s informacemi o stavu, datu, popisu a docházce. Vpravo vidíme statistiky týkající se délky trvání události, počtu revidovaných objektů a případně také počtu neopravených



Obrázek 6.8: Ukázka formulářů pro zadávání chyb, obecných ustanovení a rizik

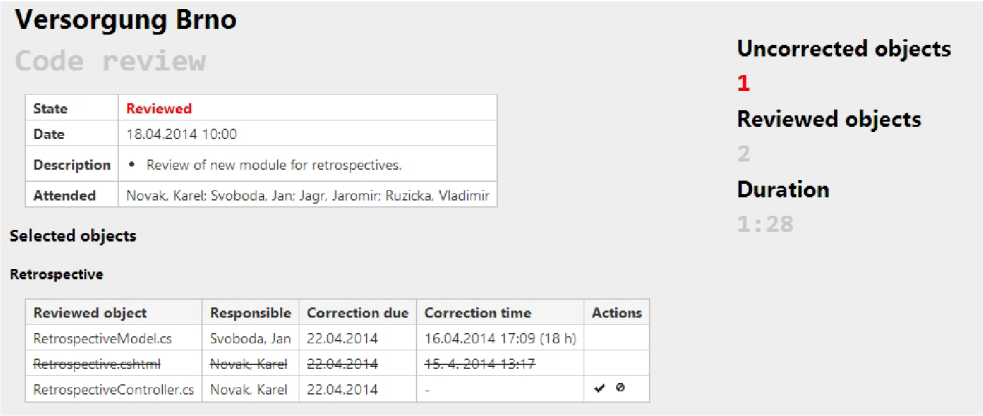


Obrázek 6.9: Ukázka kontroly rozesílaných úkolů

objektů. Pod hlavičkovou tabulkou se nalézá stěžejní část obrazovky, tedy výpis úkolů a stav oprav. Pro každý navržený objekt se vykresluje tabulka s konkrétními ovlivněnými objekty. V tabulce se zobrazí informace o jejich názvu, zodpovědné osobě, datu, do kterého má být oprava dokončena a případně datu dokončení opravy.

U neopravených objektů (na ukázce 6.10 soubor RetrospectiveController.cs) se nabízí uživateli dvě možnosti — označit objekt jako opravený nebo jako nepotřebný. U opravených objektů (na ukázce 6.10 soubor RetrospectiveModel.cs) se zobrazí čas dokončení opravy spolu s informací o délce opravy. Ta se získá jako počet pracovních hodin, které uplynuly mezi odesláním úkolu a jeho označením za dokončený. Pokud byl objekt označen za zastaralý (na ukázce 6.10 soubor Retrospective.cshtml), pak je vyjmut z celkových statistik a v tabulce se graficky označí přeškrtnutím.

Po tabulce revidovaných objektů následují popisy nalezených chyb uspořádané do záložek, výpis obecných ustanovení a výpis identifikovaných rizik. Uspořádání je stejné jako u formuláře 6.8 a proto již nenásleduje ukázka této sekce.



Obrázek 6.10: Snímek obrazovky již dokončené revize kódu

6.1.3 Statistiky

Modul Revize kódu uchovává velké množství dat, jejichž analýza může sloužit jak pro tým samotný, tak pro účely auditů apod. Z důvodu usnadnění těchto analýz nabízí modul Revize kódu vizualizaci statistik vztahujících se k ovlivněným procesům. Hlavní část statistik se zobrazuje v přehledovém pohledu a vytváří se pro pohled globální i za sprint. Další statistiky se pak zobrazují u jednotlivých revizí kódu. Grafy jsou vždy zobrazovány v pravém sloupci a jsou dále doplněny důležitými textovými informacemi například o počtu stále neopravených chyb apod. Grafy se vytváří stejně jako v původních modulech nástroje pomocí knihovny MS Chart pro .NET.

První graf (levý horní na ukázce 6.11) zobrazuje statistiku revidovaných objektů na jednotlivých revizích. Jedná se tedy pouze o navázané objekty, které nakonec byly skutečně revidovány. Do grafu se zanáší všechny revize ve stavu *revidovaná* a *uzavřená*.

Druhá statistika (levý dolní graf na obrázku 6.11) zobrazuje počty konkrétních nalezených chyb a jejich druhů. Zobrazují se všechny revize ve stavu *revidovaná* nebo *uzavřená* a každá se skládá ze tří sloupců. Ty slouží pro chyby uživatelského rozhraní, formální a obsahové.

Další graf vizualizuje doby oprav jednotlivých objektů (pravý horní na obrázku 6.11). Doba opravy se vždy počítá pouze v pracovních hodinách, tedy jako rozdíl časů počítaný pouze s osmi hodinami v pracovní dny. V grafu naleznete hodnoty pro minimální, maximální a průměrnou dobu opravy. Zobrazují se pouze revize ve stavu *uzavřená*.

Poslední vizualizací je graf s délkou trvání jednotlivých revizí kódu (pravý dolní na ukázce 6.11). Uvažuje se pouze skutečná délka trvání událostí v minutách, tedy čas od vyplnění docházky po uzavření revize. Do grafu se promítají data pro všechny revize ve stavu *revidovaná* a *uzavřená*.

6.2 Modul Retrospektiva sprintu

Druhým implementovaným modulem je Retrospektiva sprintů. Slouží pro naplánování události a následné vyplnění zápisu ze schůzky v souladu s artefaktem retrospektivy definovaným v metodice Scrum. Využívá mnoho prvků z modulu Revize kódu jako například formuláře docházky, integraci s poštovním klientem apod. Retrospektiva se může nacházet v pěti různých stavech:

Obrázek 6.11: Ukázka vytvářených statistik

Nenaplánovaná - po vytvoření sprintu

Naplánovaná - po naplánování události

Otevřená - po započetí události a vyplnění docházky

Uzavřená - po uzavření události

Wiki - speciální stav pro historickou retrospektivu obsahující pouze odkaz na interní TracWiki

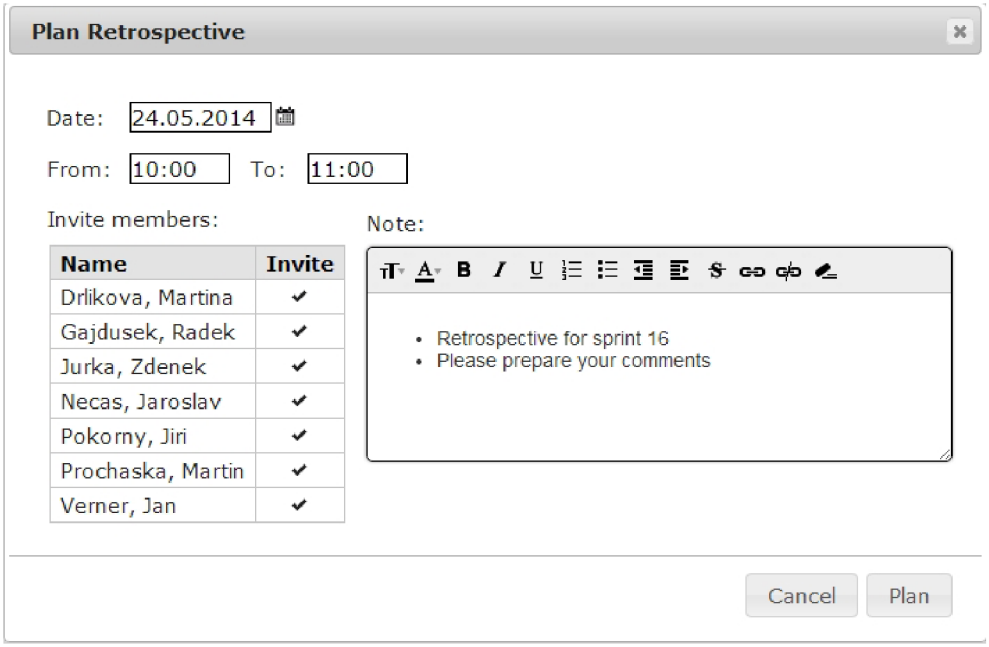
6.2.1 Plánování retrospektivy

Retrospektiva je jednoznačně identifikována týmem a sprintem, jelikož dle metodiky Scrum platí, že každý sprint má právě jednu retrospektivu. Při vytvoření sprintu se tedy zároveň vytváří také retrospektiva, která se inicializuje do stavu *nenaplánovaná*.

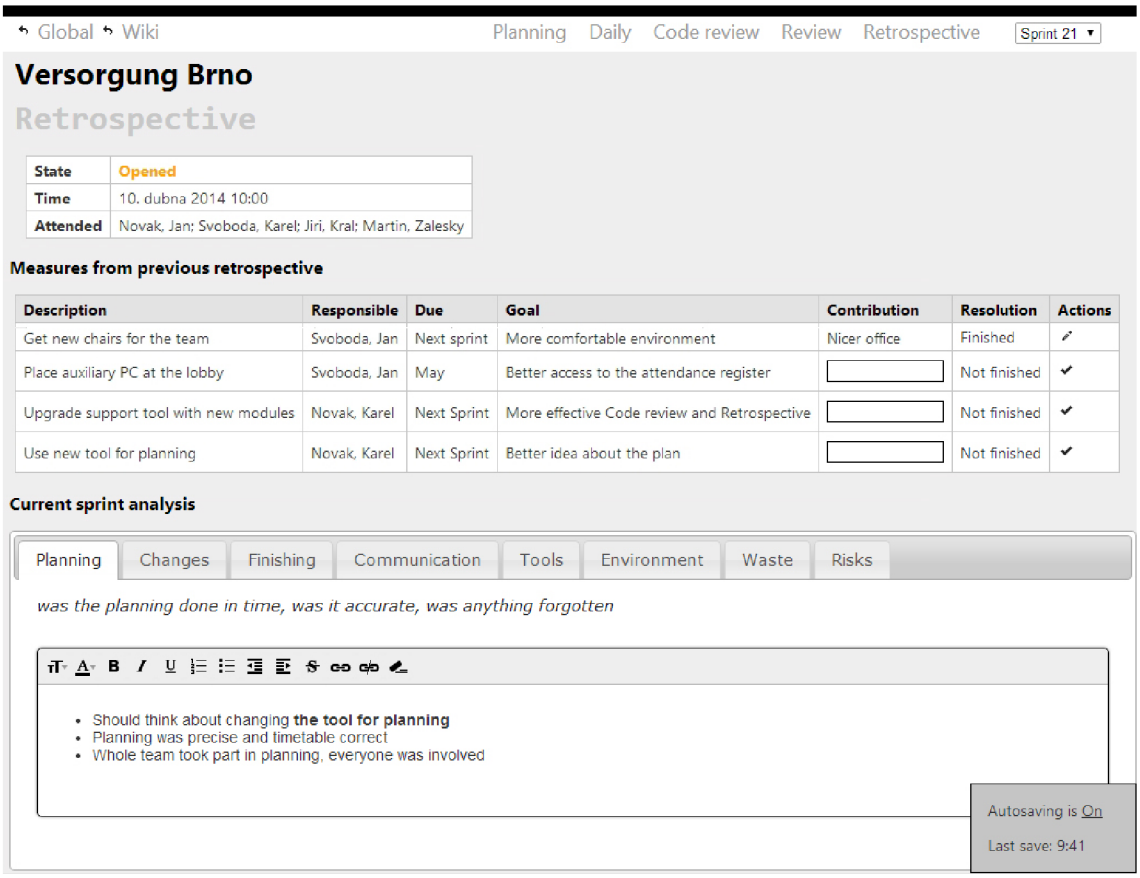
Životní cyklus retrospektivy začíná naplánováním události. K tomu slouží formulář na obrázku 6.12. Ten obsahuje vstupy pro datum a čas začátku a konce setkání, dále popis a výběr členů týmu, kterým má být zaslána pozvánka. Po odeslání všech potřebných dat se retrospektiva přepne do stavu *naplánovaná* a všem pozvaným členům se rozešle událost do poštovního klienta. Další možností je pak vložení historické retrospektivy, pro kterou se zadává pouze datum konání a URL, na kterém lze nalézt.

6.2.2 Průběh retrospektivy

Samotná událost retrospektivy začíná stejně jako revize kódu vyplněním docházky (použit je stejný formulář jako na obrázku 6.6). Po odeslání formuláře se přepne retrospektiva ze stavu *naplánovaná* do *otevřená* a zpřístupní se ovládací prvky pro vyplnění zápisu. V tento



Obrázek 6.12: Snímek formuláře plánování retrospektivy

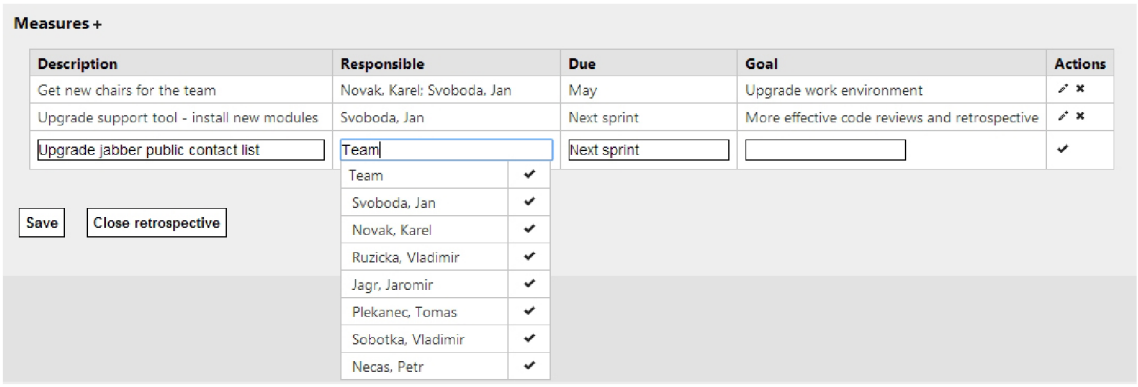


Obrázek 6.13: Obrazovka probíhající retrospektivy

moment se také začíná odpočítavat čas v ovládacím panelu fixně umístěném v pravém dolním rohu obrazovky.

První sekcí otevřené retrospektivy je revize úkolů z minulé retrospektivy. Zde se vypisují všechny úkoly z globálního katalogu, které ještě nemají nastavený příznak dokončení. V tabulce se zobrazují informace o popisu, zodpovědných osobách, datu dokončení a cíli úkolu. Pro označení úkolu za dokončený je třeba zadat jeho skutečný přínos. Dokončený úkol je pak možné zpětně vrátit do stavu nedokončený pro případné opravy zadaných vstupů.

Po revizi úkolů následuje sekce pro vložení textových komentářů k jednotlivým aspektům retrospektivy. Jednotlivá textová pole jsou řazena do záložek pro aspekty plánování, změn, dokončení, komunikace, nástrojů, prostředí, promarněného úsilí a rizik. Data v této sekci se ukládají buď po stisku tlačítka nebo pomocí automatického ukládání s minutovým intervalem.



Obrázek 6.14: Ukázka vytváření úkolů retrospektivy

Poslední sekcí průběhu retrospektivy je vytváření úkolů (na obrázku 6.14). Na základě diskuze o průběhu sprintu vznikají úkoly, které tým vyplňuje v této sekci. Zobrazují se zde také úkoly z minulých retrospektiv, které stále nebyly dokončeny a budou tak delegovány do další retrospektivy. U každého nového úkolu je nutné zadat popis, čas splnění a cíl. Dále se označí zodpovědné osoby, kterých může být více než jedna. Úkol lze po vytvoření dále editovat nebo smazat.

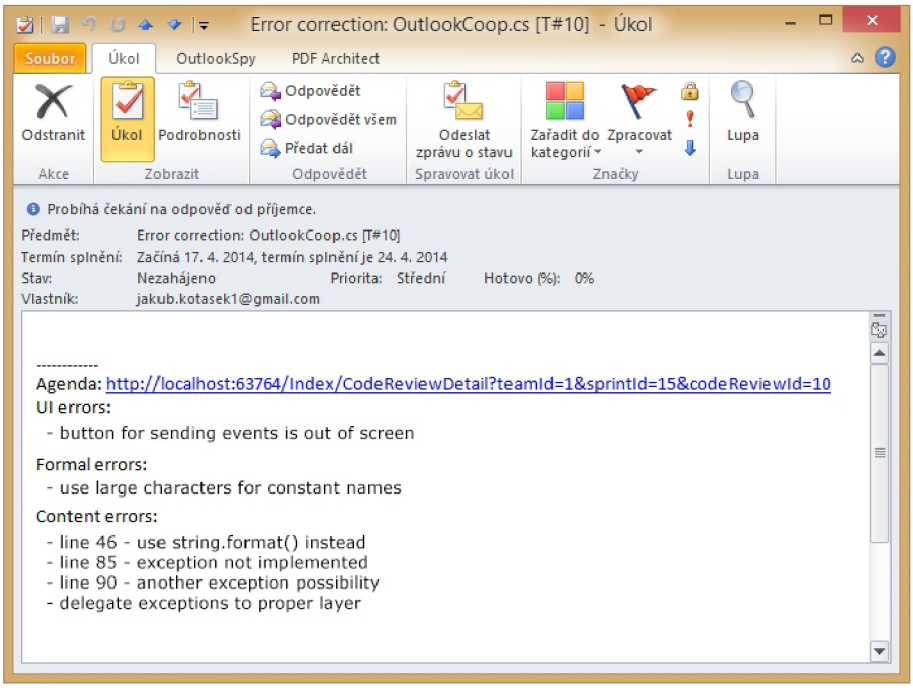
Po uzavření se retrospektiva přepne do stavu *uzavřená* a úkoly se rozešlou zodpovědným osobám do poštovního klienta. Data uzavřené retrospektivy se zobrazí již bez možnosti editace a komentáře k jednotlivým aspektům sprintu nejsou řazeny v záložkách, ale přímo vypsány. V případě nutnosti zápis dále editovat, je možné retrospektivu znovu otevřít tlačítkem pod hlavičkovou tabulkou.

6.3 Integrace poštovního klienta

Nástroj byl mimo dvou hlavních modulů rozšířen také o podporu spolupráce s poštovním klientem Outlook, která umožňuje automatické odesílání pozvánek a úkolů. Implementačně je tato funkčnost zajištěna Outlook aplikací spuštěnou na pozadí. Ten reaguje na požadavky pro odeslání úkolu nebo pozvánky a dále zajišťuje filtrování přijaté pošty a deleguje následné zpracování reakcí na pozvánky a úkoly.

Ve všech případech se pro identifikaci zpráv používá přípona předmětu následně zpracovávaná regulárními výrazy. Vzhledem k technologickým omezením poštovního klienta se

tělo zprávy zpracovává jako RTF dokument a je tedy nutné překódování vstupních dat, pro která používá celá aplikace HTML formátování.



Obrázek 6.15: Ukázka úkolu vytvořeného v modulu Revize kódu

Spolupráce s poštovním klientem se používá u modulu Revize kódu i Retrospektiva sprintu velice podobně. Nejprve se API používá pro vytvoření události a rozeslání pozvánek. Oba implementované moduly navíc nabízí možnost přeplánování události. Příjemci pak mohou na událost reagovat dvěma způsoby — přijmutí a odmítnutí. V prvním případě nebo při neodeslání odpovědi má osoba při vyplňování docházky přednastavenou volbu účastní se, ve třetím případě je pak přednastavena neúčast. Zpracování reakcí zajišťuje filtr na základě zpracování předmětu zprávy.

Další funkcí je zasílání úkolů. Úkoly jsou vytvářeny pro jednotlivé ovlivněné objekty v revizích kódu a dále pro úkoly zadané na retrospektivě. Úkoly z revizí kódu mají jednu zodpovědnou osobu a zadané nejzazší datum dokončení. Připomínka je pak nastavena na dva dny před datem dokončení. Z reakcí, které uživatel na úkol může zaslat, se filtruje pouze zpráva o dokončení úkolu. V tom případě se informace uloží do databáze k příslušnému úkolu určeném identifikátorem v předmětu zprávy. U retrospektivy se může zodpovědnost rozložit na více osob a datum dokončení se nenastavuje, pouze slovně specifikuje v těle úkolu. Reakce na úkoly z retrospektiv nejsou sledovány a Outlook démon je ignoruje.

Integrace poštovního klienta lze pomocí konstanty ve zdrojovém kódu zakázat. Při případném výpadku poštovního klienta nebo jeho nedostupnosti je aplikace stále plně dostupná, pouze se omezí funkce pro odesílání a filtraci zpráv. Pro správnou integraci je nutné instalaci klienta Outlook upravit přepsáním jedné sdílené knihovny kvůli autorizaci aplikace pro přímý přístup k funkcím pro čtení a odesílání zpráv.

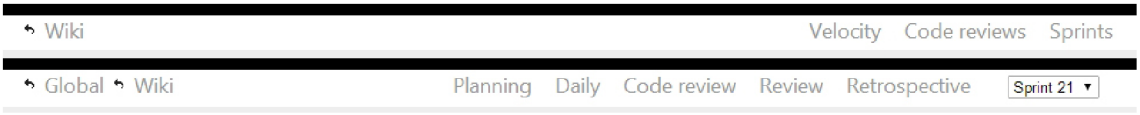
6.4 Další dílčí vylepšení

Součástí implementační části práce bylo také provedení úprav v již existujících modulech aplikace. Tyto úpravy měly za úkol především optimalizovat uživatelské rozhraní a zároveň ho sjednotit s novými prvky zavedenými v modulech Revize kódu a Retrospektiva sprintů.

6.4.1 Rozdělení globálních a sprint pohledů

Modul Revize kódu již implementuje dva možné pohledy — za sprint a globální. V rámci původních modulů aplikace existuje například modul Rychlost, který nabízí pouze globální pohled. Nabízí se tak možnost celkového rozdělení aplikace na dva pohledy, které používá modul Revize kódu.

Do globálního pohledu nově spadá modul Rychlost a Revize kódu, kde při vstupu na pohled se jako úvodní zobrazuje modul Rychlost. Sprint pohled obsahuje moduly pro plánování, revizi kódu, revizi sprintu, retrospektivu a graf zbývající práce. V případě modulu Revize kódu a Retrospektiva sprintů se v menu zobrazují také odkazy na příslušnou týmovou wiki. Pro sprint pohled je výchozí záložkou graf zbývající práce.



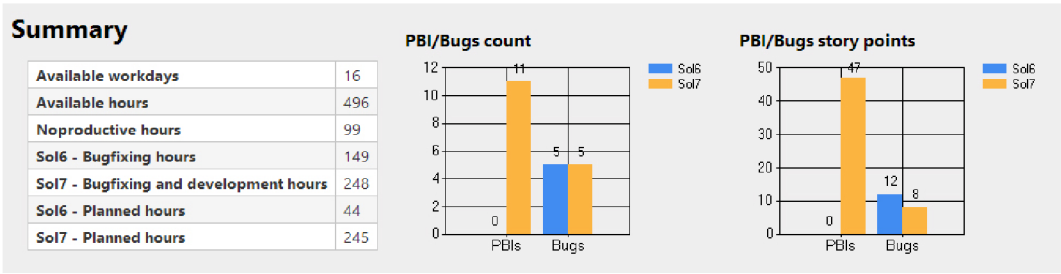
Obrázek 6.16: Navigační menu pro globální (nahoře) a sprint pohled (dole)

6.4.2 Změny uživatelského rozhraní

V modulu Plánování byly provedeny úpravy funkcí pro vkládání nedostupnosti členů týmu. Konkrétně jsou nyní funkce prováděny asynchronním voláním pro zrychlení práce s uživatelským rozhraním. Další změnou v modulu je vizualizace sumarizačních tabulek do grafů (ukázka na obrázku 6.17). Konkrétně se jedná o grafy s informacemi o počtu chyb a uživatelských scénářů. Poslední změnou v modulu pak bylo převedení textových polí pro komentář a rizika sprintu do formátování pomocí HTML.

V modulu Rychlost bylo zavedeno zobrazení do dvou sloupců. Nově se tak minimalizovala nutnost skrolování a graf i tabulka pro statistiky splnění naplánovaných úkolů se zobrazují vedle sebe.

V modulu Revize byly převedeny data ze sumarizační tabulky do grafu. Informace o splněných a naplánovaných uživatelských scénářích se tak nově vizualizují. Další změnou v tomto modulu bylo zavedení HTML formátování pro textová pole rizik a komentáře.



Obrázek 6.17: Ukázka změn v modulu Plánování

6.5 Testování

Důležitou součástí implementace bylo také průběžné a závěrečné testování aplikace. K tomu sloužilo několik různých prostředků. Především se jedná o zavedené testovací prostředí, které pomocí virtualizace přiblížilo testování aplikace reálnému systému na oddělení Corporate Technology v Brně. Proběhla také revize kódu implementovaných modulů a nakonec byla aplikace nasazena v produkčním prostředí.

6.5.1 Testovací prostředí

Jako testovací prostředí sloužil virtualizovaný server spravovaný pomocí software VMWare Workstation. Pro server byl použit operační systém Windows Server 2008 R2, kterému byly doinstalovány role IIS (Internet Information Services) a Exchange serveru. IIS server sloužil pro provoz samotné webové aplikace a server Exchange byl nutný pro testování funkcí pro integraci poštovního klienta. API poštovních klientů bylo testováno na verzi Outlook 2007 a Outlook 2010, nicméně výrobce udržuje API zpětně kompatibilní až do verze 2003.

6.5.2 Testovací scénáře

Vzhledem k technickým specifikacím a časové náročnosti automatizování testů uživatelského rozhraní byla zvolena kombinace regresních a „monkey“ testů se scénáři pokrývajícími co největší část aplikace. Regresní testy obsahují testovací scénáře a „monkey“ testy probíhají náhodně ve snaze uvést aplikaci do nekonzistentního stavu.

Důležitým testovacím nástrojem integrovaným přímo do vývojového prostředí Visual Studio jsou jednotkové testy. Ty umožňují rychlé otestování kritických částí aplikace, bohužel ale nejsou schopny kontroly externích aplikací — v našem případě poštovního klienta Outlook. Většina testů proto byla sestavena ve formě regresních testů ve virtualizovaném prostředí.

Regresní testy modulu Revize kódu

Scénář tvorby historické revize:

1. Vyber globální pohled — nejsou dostupné ikony pro vytváření revize kódu.
2. Vyber sprint pohled — jsou dostupné ikony pro vytvoření revize kódu a pro vytvoření historického zápisu.
3. Zvol historický zápis — zobrazí se modální dialogového okno se vstupy pro datum a odkaz.
4. Klikni do vstupu pro datum — zobrazí se kalendář omezený daty aktuálně zvoleného sprintu.
5. Zvol datum a zadej odkaz — případný odkaz ve špatném formátu je odmítnut, pro zadání data funguje kalendář.
6. Odešli formulář — v seznamu revizí se vytvoří nová revize ve stavu *wiki* a možností smazání.

Scénář tvorby a inicializace události:

1. Zkontroluj globální i sprint pohled pro poslední sprint v databázi — zobrazí se upozornění na nedostatek dat a nezobrazí se žádné statistiky.
2. Vytvoř v globálním pohledu objekt k revizi — objekt se zobrazí v globálním pohledu, ve sprint pohledu bude stále upozornění na nedostatek dat.
3. Edituj objekt k revizi — změny se projeví v globálním pohledu.
4. Vytvoř revizi kódu u posledního sprintu bez zvolených objektů k revizi — rozešle se pozvánka v poštovním klientu a revize se zobrazí ve stavu *naplánovaná* v globálním i sprint pohledu.
5. Otevři revizi kódu a přiřaď dříve vytvořený objekt k revizi — objekt k revizi se zařadí do tabulky navrhované objekty.
6. Odmítni pozvánku v poštovním klientu — u detailu revize se k příslušnému členu týmu přednastaví neúčast.

Scénář průběhu revize kódu:

1. Vyplň docházku — zpřístupní se vytváření zápisu a začne odpočet času.
2. Zapni automatické ukládání a zvol objekt k revizi — každou minutu se ukládají data a objekt k revizi se přesune do tabulky zvolených.
3. Přidej ovlivněný objekt, uprav ho, smaž a přidej znovu — změny se vždy projeví také u záložek s textovými poli.
4. Vyplň textová pole pro ovlivněný objekt, přiřaď zodpovědné osoby a vyplň obecná ustanovení a rizika — po automatickém uložení a obnovení stránka data zůstavají.
5. Ukonči revizi kódu — zodpovědným osobám se rozešlou úkoly do poštovního klienta, v globálním i sprint pohledu budou obecná ustanovení a rizika.
6. Označ úkol v poštovním klientu za hotový, pak další úkol ručně v detailu revize a další jako zastaralý — změny se projeví v detailu i statistikách.
7. Po označení všech úkolů za hotové se přepne revize do stavu *uzavřená* a data se projeví ve statistikách.

Regresní testy modulu Retrospektiva sprintů

Scénář tvorby historické retrospektivy:

1. Zvol sprint s nenaplánovanou retrospektivou — jsou dostupné volby pro vytvoření historické a normální retrospektivy.
2. Zvol historický zápis — zobrazí se modální dialogového okno se vstupy pro datum a odkaz.
3. Klikni do vstupu pro datum — zobrazí se kalendář omezený daty aktuálně zvoleného sprintu.

4. Zvol datum a zadej odkaz — případný odkaz ve špatném formátu je odmítnut, pro zadání data funguje kalendář.
5. Odešli formulář — pro zvolený sprint se zobrazí informace o historické retrospektivě spolu s funkčním odkazem.

Scénář tvorby a inicializace události:

1. Zvol sprint s nenaplánovanou retrospektivou — jsou dostupné volby pro vytvoření historické a normální retrospektivy.
2. Zvol normální retrospektivu — zobrazí se modální dialogové okno se vstupy pro datum, čas, poznámku a výběr pozvaných členů.
3. Výběr data je omezen zleva dnešním datem.
4. Nevybírej všechny členy týmu.
5. Odešli formulář — vytvoří se událost a odešle do poštovního klienta, zobrazí se retrospektiva ve stavu *naplánovaná* s možností přeplánování.
6. Odmítni pozvánku v poštovním klientu — u detailu retrospektivy se k příslušnému členu týmu přednastaví neúčast.

Scénář průběhu retrospektivy:

1. Vyplň docházku — zpřístupní se vytváření zápisu a začne odpočet času.
2. Zapni automatické ukládání — každou minutu se ukládají data a aktualizuje se čas posledního uložení.
3. Označ úkol z minulé retrospektivy za hotový bez vyplnění přínosu — zobrazí se chybová hláška, přínos je nutné vyplnit.
4. Označ úkol z minulé retrospektivy za hotový s vyplněním přínosu — úkol se označí za dokončený a skryje se v sekci pro úkoly do další retrospektivy.
5. Smaž úkol z minulé retrospektivy — úkol se skryje v sekci úkolů z minulé retrospektivy a pro další retrospektivu.
6. Vytvoř úkol pro další retrospektivu — funguje volba více zodpovědných osob a je nutné vyplnění všech polí.
7. Uzavři retrospektivu — úkoly, které nemají příznak dokončení se rozešlou příslušným zodpovědným osobám.

6.5.3 Nasazení v produkčním prostředí

Po implementaci hlavních částí modulu následovala také revize kódu samotným vytvořeným modulem. Revize kódu proběhla za účasti konzultanta společnosti a vybraného týmu a měla za úkol ověřit jak funkčnost, tak zdrojový kód. Před nasazením do produkčního prostředy tedy byl proveden celý cyklus modulu Revize kódu.

Po důkladném otestování ve virtuálním prostředí pomocí regresních a „monkey“ testů a po revizi kódu byl nástroj nasazen v produkčním prostředí oddělení Corporate Technology

v Brně. Při provozu bylo nalezeno a odstraněno několik dalších chyb, ale především byly provedeny úpravy na základě praktického využití aplikace tak, aby co nejvíce odpovídala potřebám týmů oddělení v Brně.

Kapitola 7

Zhodnocení a další vývoj

Práce splnila své hlavní cíle a sice ušetřit čas členů týmu a zvýšit kvalitu ovlivněných procesů. Pro budoucí vývoj se pak nabízí další možné moduly, které by ještě více optimalizovaly administrativní zátěž a přinesly další časovou úsporu.

7.1 Přínos aplikace

Hlavními cíli práce bylo ušetřit čas optimalizováním administrativního zatížení týmu a dále zvýšit kvalitu implementovaných procesů metodiky Scrum. Toho se podařilo dosáhnout automatizováním některých částí procesů revize kódu a retrospektivy a dále použitím globálních katalogů pro zdůraznění pozitivních dopadů na tým. Aplikace je již nyní plně nasazena a začleněna do vývojového cyklu týmů na oddělení Corporate Technology společnosti Siemens v Brně. Dle oborného odhadu vedoucího oddělení bylo díky novým modulům dosaženo 15% úspory času u ovlivněných procesů.

U modulu Revize kódu byl zaveden globální katalog objektů určených k revizi. Závěry revizí kódu jsou mimo nalezené chyby také obecná ustanovení a rizika. Tyto položky se v minulosti uchovávaly pouze u textových zápisů na wiki. Modul nyní poskytuje globální katalog těchto položek a zobrazuje je na viditelném místě dle globálního nebo sprint pohledu. Nemělo by se tak již stávat, že například obecné ustanovení o stylu názvů proměnných bude dodržováno jenom některými členy týmu, jelikož všichni nebyli přítomni na revizi kódu.

Revize kódu dále přináší úsporu času díky automatizování spolupráce s poštovním klientem. V minulosti bylo nutné při pořádání události vytvářet ručně jak šablonu pro textový zápis, tak Outlook událost. Nově se pouze vyplní formulář s datem, časem a pozvanými členy týmu. Hlavní úspory času ale bylo dosaženo automatizováním zadávání a sledování oprav nalezených chyb. Dříve bylo nutné po skončení revize vytvořit jednotlivé Outlook úkoly s nalezenými chybami a následně sledovat jejich stav dokončení. Díky modulu se úkoly automaticky vytvoří, vyplní i rozešlou a následně aplikace sleduje jejich stav a přenáší stav přímo do databáze.

Přínos modulu Retrospektiva sprintů se nachází v podobných funkcích jako u předchozího modulu. Automatizováno je vytváření události i její rozeslání a sledování odpovědí na pozvánky. Modul zavádí globální katalog úkolů z retrospektiv, díky čemuž nabízí automatické předvyplnění zápisu nedokončenými úkoly z minulých retrospektiv. Tvorba šablony retrospektivy se v minulosti prováděla ručně a automatizace přináší asi nějvětší úsporu času v tomto modulu. Novou funkčností je rozesílání úkolů po skončení retrospektivy po-

mocí poštovního klienta. Tento krok se neprováděl kvůli časové náročnosti a nyní jsou tak úkoly pro členy týmu viditelnější.

Dílčí rozšíření v již existujících modulech přinesla především další optimalizaci uživatelského rozhraní, díky kterému bylo dosaženo ještě větší přehlednosti a rychlejší odezvy. Při tvorbě uživatelského rozhraní v celé práci vzniklo také mnoho kódu použitelného na dalších projektech a konkrétně například funkce pro selektivní seznamy budou vydány jako rozšíření pro framework jQuery.

7.2 Možný další vývoj

Některé moduly vhodné pro implementaci byly představeny již v kapitole 4.2. Největší přínos z hlediska časové úspory by oddělení Corporate Technology přinesl modul Týdenního setkání. Dalším vhodným modulem je Grafická nástěnka. Její implementace by přinesla další podporovanou agilní metodiku Kanban. Ta se v současnosti hojně využívá a většina profesionálních nástrojů ji podporuje. Další potenciální modul vycházející z trendů profesionálních nástrojů by mohl nabídnout konfigurovatelnou obrazovku určenou speciálně pro promítání na pracovišti. Momentálně se na pracovišti promítají přímo obrazovky modulů a zavedení konfigorvatelného modulu by mohlo přinést další zpřehlednění.

U modulu Revize kódu by bylo možné posílit napojení na verzovací nástroj a vývojové prostředí. Nabízí se například funkčnost, která by umožnila přímou propagaci nalezených chyb také do Team Foundation Server. Dále by bylo možné modul rozšířit o přímé hypertextové odkazy do vývojového prostředí. V tomto případě by bylo nutné implementovat rozšíření pro Visual Studio, které by uživatelům nabídlo protokol pro vytváření odkazů do souborů, tříd nebo přímo řádků implementace. Vzhledem k tomu, že modul Revize kódu není kriticky závislý na zbývající aplikaci, bylo by možné exportování a integrace například jako rozšíření pro profesionální nástroj JIRA.

Další oblastí pro rozšíření je samotné jádro aplikace. V současnosti je aplikace zcela otevřená a zavedení autentizace a autorizace uživatelů by mohlo přinést splnění dalších podmínek nutných pro nasazení na dalších odděleních. Jádro aplikace by v tomto případě mohlo být rozšířeno také o konfigurační rozhraní nabízející správu týmů, sprintů a členů týmu. Budoucí rozvoj aplikace by také měl brát v úvahu distribuované týmy a rozsáhlé týmy skládající se z několika Scrum týmů najednou. Všechny tyto úpravy by měly vést k jednoduššímu nasazení aplikace u dalších týmů.

Výhodou modulu Revize kódu je možnost jeho využití i bez implementované metodiky Scrum. Modul lze zpřístupnit samostatně bez Team Foundation Server nebo poštovního klienta Outlook, a proto je možné nasazení také na všechny týmy, které v současnosti teprve plánují zavést proces revize kódu nebo by ho rády automatizovaly.

Kapitola 8

Závěr

Práce se zabývá agilním vývojem software s důrazem na metodiku Scrum. Cílem bylo rozšíření nástroje pro podporu agilního vývoje používaného v oddělení Corporate Technology společnosti Siemens, s.r.o. v Brně. Účelem tohoto rozšíření bylo optimalizovat administrativní zátěž a zkvalitnit procesy implementované metodiky Scrum. Pro splnění specifických požadavků oddělení se práce inspirovala v profesionálních nástrojích jako JIRA, Team Foundation Server nebo ScrumWorks Pro.

Na základě analýzy profesionálních nástrojů a současného stavu na oddělení bylo identifikováno několik možností pro rozšíření původního nástroje. Z možných rozšíření byly po konzultaci se zástupci oddělení zvoleny k implementaci dva moduly s největším potenciálním přínosem pro oddělení. Modul Revize kódu slouží pro pořádání události, vytváření zápisu a kontrolu oprav skupinové revize kódu. Modul Retrospektiva sprintů pokrývá artefakt retrospektivy metodiky Scrum a zajišťuje pořádání události, vytváření úkolů a tvorbu zápisu. Dále bylo implementováno také několik dílčích vylepšení původních modulů nástroje.

Rozšíření nástroje přineslo dle kvalifikovaného odhadu vedoucího oddělení asi 15% úsporu času u ovlivněných procesů. Té bylo dosaženo především automatizací pořádání událostí retrospektivy a revize kódu a automatickou kontrolou oprav nalezených v revizích. Kvalita ovlivněných procesů se navíc zvýšila zdůrazněním pozitivních dopadů na tým, čehož bylo dosaženo díky globální správě revidovaných objektů, závěrů revizí i závěrů retrospektiv.

V současné době jsou implementované moduly nasazeny v produkčním prostředí oddělení Corporate Technology v Brně. O nasazení aplikace projevily zájem také další oddělení společnosti Siemens, především pak o modul Revize kódu, který je možné nasadit na týmy nepoužívající Scrum, Team Foundation Server nebo poštovního klienta Outlook. Výsledky řešení diplomové práce byly prezentovány na studentské soutěžní konferenci STUDENT EEICT 2014. Práce se umístila na třetím místě v kategorii Informační systémy a setkala se také s velice pozitivními ohlasy komisařů - zástupců firem. Diplomová práce splnila všechny body zadání a její nadprůměrný přínos dokazuje široký zájem o vytvořené moduly i úspěch ve studentské soutěži EEICT.

Literatura

[1] Rally Software — Enterprise Proven Agile [online]. `http://www.rallydev.com/`, [cit. 2014-02-05].

[2] Assembla, I.: Task & Issue Management, Collaboration — Assembla [online]. `https://www.assembla.com/home`, [cit. 2014-02-06].

[3] Atlassian: JIRA – Atlassian [online]. `http://www.atlassian.com/software/jira`, [cit. 2014-01-07].

[4] Battat, S.: Team Foundation Server – Agile Project Management Using TFS 2012 [online]. `http://msdn.microsoft.com/en-us/magazine/dn189203.aspx`, [cit. 2014-01-08].

[5] Beck, K.: *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 1999, iSBN 078-5342616415.

[6] Cockburn, A.: *Crystal Clear: A Human–Powered Methodology for Small Teams*. Addison-Wesley Professional, 2004, iSBN 978-0201699470.

[7] Cockburn, A.: *Agile software development: the cooperative game*. Upper Saddle River, NJ: Addison-Wesley, 2007, iSBN 0-321-48275-1.

[8] Coffin, D., Rod; Lane: *A Practical Guide to Seven Agile Methodologies [online]*. DevX.com, [cit. 2014-01-07].

[9] CollabNet, I.: ScrumWorks Pro -– Powerfull, Agile project management for Scrum [online]. `http://www.collab.net/products/scrumworks`, [cit. 2014-01-08].

[10] Kolektiv autorů: *Survey of Agile Tool Usage and Needs*. Agile Conference (AGILE), 2011, iSBN 978-1-61284-426-8.

[11] Kolektiv autorů: Manifesto for Agile Software Development [online]. `http://agilemanifesto.org/`, [cit. 2013-12-28].

[12] Kolektiv autorů: Scrum Guide^{TM} [online]. `http://www.scrum.org/Scrum-Guide`, [cit. 2013-12-28].

[13] Larman, C.: *Agile and iterative development: a manager's guide*. Boston: Addison-Wesley, 2004, iSBN 0-13-111155-8.

[14] Schwaber, M., Ken; Beedle: *Agile software development with Scrum*. Upper Saddle River: Prentice Hall, 2002, iSBN 0-13-067634-9.

[15] ThoughtWorks, I.: Mingle – Agile project management software [online]. http://www.thoughtworks.com/products/mingle-agile-project-management, [cit. 2014-02-05].

[16] VersionOne, I.: Agile Project Management Software — VersionOne [online]. http://www.versionone.com/, [cit. 2014-02-06].

Dodatek A

Obsah CD

Na přiloženém CD se nachází následující adresářová struktura:

/thesis/ - text práce ve formátu PDF i LaTeX

/src/ - zdrojové kódy aplikace psané v jazyce C#

/demo/ - offline demoverze aplikace

/readme/ - návod a potřebné soubory pro integraci poštovního klienta Outlook